# DRAGON USER

*The independent Dragon magazine*

May 1987

## Contents



## Editorial

**THE 6809 Show was small but busy, and I had the opportunity to match up quite a few names and faces. I left late and still didn't get to talk to all the people I wanted to talk to, concentrating as I was on the ones who are hardest to get hold of, and could quite happily have kept myself occupied till wrapping up time if circumstances had permitted.**

**Cheers again to everyone who came along, and special thanks to Jenny Pope who organised the show, Jason for answering tricky questions and scavenging everything he could lay his hands on, Helen Perry and David Osen for fetching and carrying and personfully personing the stall in the afternoon, Tim Lomas of NDUG for offering me a pint which I didn't have time to drink, Alan Preston of R & AJ Preston for contributing some *Boulder Crashes* above the call of duty to help us out of an embarrassing situation, and above all to EVERYONE WHO RE-SUBSCRIBED TO DRAGON USER. In the heat of enthusiasm or the depths of despair it is easy to forget that this year you've *got* to remember your sub, or we won't know you're there, and we won't be. Don't delay.**


**Telephone number**
(All departments)
437-4343

**Editor**
HELEN ARMSTRONG

**Production Editor**
BARBORA HÁJEK

**Associate Editor**
JOHN COOK

**Editorial Secretary**
ANNE MARIE O'DWYER

**Advertisement Manager**
DAVID OSEN

**Administration**
GABRIELLA PISCINA

**Marketing Manager**
HELEN PERRY

**Managing Editor**
PETER WORLOCK

**Publishing Director**
JENNY IRELAND

**Subscriptions**
UK £14 for 12 issues
Overseas (surface) £20 for 12 issues
ISSN 0265-177. Telex: 296275
*Dragon User*, 12/13 Little Newport Street, London WC2H 7PP
US address: c/o Business Press International, 205 East 42nd St, New York, NY 10017
Published by Scot Press Ltd.



Typesetting and Production by Artext Limited, London NW1.
Printed by Headley Brothers Ltd. Ashford, Kent
Registered at the Post Office as a newspaper.
Dragon and its logo are trademarks of Eurohard Ltd.


**How to submit articles**

The quality of the material we can publish in *Dragon User* each month will, to a very great extent depend on the quality of the discoveries that you can make with your Dragon. The Dragon computer was launched on to the market with a powerful version of Basic, but with very poor documentation.

Articles which are submitted to *Dragon User* for publication should not be more than 3000 words long. All submissions should be typed. Please leave wide margins and a double space between each line. Programs should, whenever possible, be computer printed on plain white paper and be accompanied by a tape of the program.

We cannot guarantee to return every submitted article or program, so please keep a copy. If you want to have your program returned you must include a stamped addressed envelope.

# Letters

This is the chance to air your views — send your tips, compliments and complaints to Letters Page, *Dragon User*, 12-13 Little Newport Street, London WC2H 7PP.

## Just a thought

THE shiny, black, new, state of the art £229.95 Z88 Cambridge Computer masterminded by Clive Sinclair is advertised as being able to run its ROM word processor with RAM of 'around 15K available'.

My lovely, lowly, out-dated, bottom-of-the-range Dragon 32 runs its cartridge *Super Writer II* word processor with 22870 bytes (22.33K) of available RAM — or 28928 bytes (28.25K) if the graphics capability is DUMPed.

*Mike Hosken*
*Hazelwood*
*Pinford End*
*Hawstead*
*Bury St. Edumunds*
*Suffolk IP29 5NU*

**And it has a decent screen size, and you can't fit it in a briefcase, which means that nobody can get on at you to work on the train.**

## More OS-9 news

In answer to recent letters, I can report that it is possible to obtain the original Dragon OS-9 plus the manual from: Karsten G. Ludwig Data Consulting, Dreieichstrasse 27, D-6057 Dietzenbach, Germany.

Alternatively it is possible to order the Tandy Color Computer OS-9 which includes also the assembler, editor and a thick pile of manuals. CoCo OS-9 does not boot straight on the Dragon: you have to replace the boot files, disc, printer and keyboard and ACIA drivers with the versions for the Dragon OS-9. The address of Tandy is: Radio Shack, 300 One Tandy Center, Fort Worth, Texas 76102, USA.

OS-9 users in the UK should contact the NDUG's OS-9 guru, Jason Shouler, at 70 Victoria Road, Parkstone, Poole, Dorset BH12 3AE.

We here in Finland have Dragon OS-9 patches which allow a terminal to be used through the serial port at 9600 baud, 7 + 2 bits, no parity, and actually use the 6msec stepping rate on the upper drive, and a 'Europatch' that allows ASCII 91-93 and 193-195 to appear in path names.

Another useful address is the US OS-9 user Group, Dale Puckett, PO Box 7586, Des Moines, Iowa 50322, USA. Membership was $25 a year at the last count.

Commercial OS-9 software is available from Computape, 27 Coombe Road, Southminster, Essex CM0 7AH, and in the USA through Tandy or from Frank Hogg Lab. Inc., The Regency Tower, Suite 215, 770 James St., Syracuse, NY 13203, USA.

*Martin Vermeer*
*Dragon User Club, Finland*
*Punahilkantie 4 B 14*
*SF-00820 Helsinki 82*
*Finland*

**THIS letter has been in the system for a while, but the information is useful and probably still up to date. If in doubt, write to them and enquire.**

**Every month we will be shelling out a game or two, courtesy of Microdeal, to the reader/s who send the most interesting or entertaining letters. So send us your hints and your opinions, send us your hi-scores and suggestions. Send us your best Dragon stories. What d'you think we are, mind readers?!**

EXTRA PUFF

## What a pain!

DO I detect some longstanding animosity between Messrs. Hitchman and Orbaum? Is it necessary to submit *Dragon User* readers to this kind of petty carping? Surely the job of a games reviewer is to review games and not to try to be some kind of literary superwit? I have a copy of Fire Force and find it very interesting. As a humble programmer myself I can imagine the pain this review will have inflicted on the poor author who must have spent many hours writing the program.

I was under the impression that DU was designed to promote the Dragon, but this type of journalism is contrary to this ideal. If you discourage the authors and Dave Hitchmans of this world (and there are frighteningly few) you destroy the Dragon. Long live DU, long live DH and return JO to full time accountancy. If you need a replacement, I'm available!

*Hugh Fowler, 1 Victoria Road, Irlam, Manchester*

**MR. Fowler, your letter touches on a number of important points. The irony of a reviewer's lot is that he or she does not have to be the instigator of a long term feud in order to find him/herself participating in one. I am not going into a list of namechecks, but Jason Orbaum was not at all the only person who was unhappy about *Fire Force*. Now, we haven't printed letters from everyone who has written, because, as you so rightly point out, it should not be DU's policy to pick on suppliers for individual mistakes or misunderstandings. But it is our job to give a voice to readers with negative views as well as positive, whenever those views are fairly supported. If a number of readers have bought a game which they feel is poor value, then their views must be put forward. It is purely political to call an opinion 'carping' because it is negative.**

**For instance, one frequent complaint about *Fire Force* is that the collision detection is inconsistent. Some people are very irritated by this factor in a game, while others ignore it. If a programmer does not take this spread of opinions (which is as old as computer gaming) sufficiently into account then he is going to have a number of seriously disappointed customers, no matter how perfect the programming is, or how long it took. This is a fact. Games don't come free, people are parting with their hard earned readies.**

**The controversy about *Fire Force* is bound to continue. Jason believes that facets of the game which make it difficult to play are 'bugs'. Dave Hitchman stresses that they are part of the game's design. A 'bug' is a programming *or a critical design fault* in a program, and there comes a point where it becomes a matter of opinion whether you are dealing with a mistake or a challenge. Jason Orbaum looks at games closely, and rarely makes unsupported statements, however he dresses them up.**

***Dragon User* would very quickly lose credibility if it only talked about the good news. We've put Jason's point of view, allowed Dave an extra month to get a reply to us before publishing, and run other reviews putting pro and con points.**

**Jason's review has not frightened Dragon users away from the game in droves — Quickbeam report good sales since the review appeared. Which demonstrates what we already know, that readers are prepared to use their own skill and judgement. On that subject, let me say again that anyone who wants to send a review in is welcome to do so. We can only print a selection, but if you want to show DU your mettle, do what writers do — write something!**

## Each way, John?

HAS anyone noticed that if you hold down the keys J, O, H and N while playing *Moon Cresta*, a message comes up on the screen.

Maybe other software houses will take a tip from Incentive and have a Dragon and Tandy colour version of the game on the game cassette.

*Paul Cartier*
*110 Mawney Road*
*Romford, Essex*

## Video titles

CAN anyone who has used *Rainbow Writer* for making video titles please let me know the way they went about it?

*D. W. Clarke*
*61, Barnsfold, Fulwood,*
*Preston, Lancs., PR2 3EU*

PS This would make a useful article for a future magazine.

**IS anyone interested in taking up one or both or these gauntlets?**

# Stuck up a tree

With reference to *Fire Force*, Jason Orbaum's review was sarcastic and does have one or two inaccuracies.

The gun *does* work properly when you get the knack of it, and the game is not 'completely bug ridden'.

The only drawback that I can see is getting stuck in trees, and you cannot always jump over them by wiggling the joystick backwards and forwards.

*Fire Force* is an excellent piece of programming with good graphics and sound.

A plea: does anyone know the Passcode to the game and what it does? I can get past the Authorisation code to the Passcode but no further!

*Douglas J. Field*
*14 Lawnswood*
*Hessle*
*N. Humberside*
*HU13 0PT*

# Next, please

I READ the review of *Fire Force* with total agreement. After receiving the game on Christmas Day, I loaded it with anticipation of seeing a 'commando' style game at last for my Dragon.

At first sight the game looked very good. Excellent sound and graphics, but when it came to the gameplay I was very disappointed.

The figures move jerkily about, and it is almost impossible to last over fifteen seconds with one man.

I do not agree that it is a challenge 'for the serious games player', more an impossible task for anyone.

I am not criticising Quickbeam for the hell of it, *Shaolin Master* is one of my favourite. I look forward to seeing Six-Aside-Soccer hoping it is to Shaolin standards.

*David Humphreys*
*Coed Helen*
*Lluyndyrys*
*Pwllheli*
*Gwynedd*
*LL53 6N6*

PS Please expand the Expert's Arcade Arena, perhaps by getting rid of some of the listings or serious stuff.

# Telewriter disable

OCCASIONALLY, with *Telewriter*, it is advantageous to disable the program's control over the number of printed characters per line. An example would be using it with a printer that has proportional spacing and better built-in justification capability than the word processor's. To use such features *Telewriter* must be tricked into thinking each paragraph is a single line. This required more than just a ⟨CLEAR-D⟩ which only disables wordwrap on the screen. For the text to be printed properly the (C)haracters per line is set to 128 or larger and all carriage returns from the start of the paragraph to the end are removed. The margins and line lengths are controlled by printer escape codes rather than *Telewriter* (M)argin and (C)haracter commands.

As writing and editing without wordwrap is cumbersome I leave it enabled until the text is ready to be printed. Wordwrap is then disabled with a ⟨CLEAR-D⟩. Carriage returns, which are not displayed, are found by positioning the cursor at the start of the paragraphs and executing a ⟨CLEAR-right arrow⟩. The process is repeated until the only carriage return left is at the end of the paragraph. Carriage returns terminating blank and single lines are left in place. At this point one must be careful not to use the ⟨CLEAR-A⟩, align command, as it will reinsert all the carriage returns which were removed.

Printer escape codes for setting left and right margins are defined at the start of the text. Also the (C)haracter and (M)argin commands, eg 'M0 C128', are defined there. The (M)argin can be set to a non-zero value for offsetting short lines. One drawback to this whole procedure is that *Telewriter* no longer keeps track of the number of printed lines. So page breaks and numbering will have to be done manually printer escape codes.

*Allan Worrest*
*5144 Holdrege Street*
*Lincoln, Nebr. 68504, USA*

# Stylistic changes

THANK you at last for an article on the OS-9 operating system, and in particular how to modify *Stylograph* to remove the dreaded line feed and other improvements.

However, I must take either you or Mr. Rothery to task on the inaccuracies in this otherwise excellent article. As it stands, any attempt to follow the 'change a byte' instructions merely resulted in error after error. I only tried to alter 'Stylo' but discovered the following:

'*To change a byte*'

1) DEBUG # 2k is ok
2) $load /d1/stylo is ok
3) L stylo is ok
4) The printout was as expected is ok
5) Dot command is ok
6) There was a . ahead of the = in my copy, there should be no dot here, just enter =00 (or whatever).
7) Q is ok
8) This line is quite wrong, it should read:
SAVE /D1/STYLO.mod STYLO (See pages 7 to 44 of the OS-9 manual)
9) Yet another error in this line, which should read:
VERIFY U ⟨/D1/STYLO.mod ⟩/D1/STYLO (see pages 7 to 56 of the OS-9 manual)
Note: Before carrying out step 9 it is necessary to have deleted the original STYLO from disc /D1, otherwise call the file "STYLO.ver" or something and rename it later.
10) ATTR /D1/STYLO e pe is ok subject to the note above.

I didn't try any more of the items except the remedy for SPELL. I discovered that the empty file was only one of the problems and as far as it went the article solved this. However, for some reason my copy of SPELL will not rewrite the text file, nor will it display words in context after the review process. Attempts to make it do so simply result in an error message.

I thank Mr. Rothery for an otherwise fine article and I have already written to the OS-9 User Group for details of membership.

*D. L. Jones*
*Corbetts*
*Queens Road*
*Ilkley*
*W. Yorks*
*LS29 9QJ*

**Here follow edited highlights from David Rothery's reply to Mr. Jones and to DU:**

"The errors were mostly caused, I think, but the precise way OS=9 instructions must be given if they are to work, and typesetting that lot with every dot and space correct would have been very difficult. However, I take full responsibility for transposing the SAVE instruction and forgetting to tell people to delete the old file.

"However, all the locations to be changed should be accurate, and with a litle reference to the manual most people would still find the article of use.

"With reference to the problem with SPELL, did Mr. Jones transfer it onto a separate disc? If so you will need to copy DEL, RENAME and possibly COPY because SPELL uses these to change the file.

"To change a byte should have read:
DEBUG # 2k
$load /d1/stylo
L style
. 7F52
=00
make further changes the same way
Q
SAVE /d1/stylo.mod stylo
DEL /d1/stylo
VERIFY U ⟨/d1/stylo.mod ⟩/d1/stylo
ATTR /d1/stylo e pe

The comments on each line were correct, as I hope were all the tables of alterations.

"Also, in case anyone thinks I am repeating myself, myself the letter on the same subject was an alternative, and not intended to be published as well!

"I have had lots of nice letters and phone calls about my article OS-9 Blues, thanks a lot for publishing it."

*David Rothery*
*1 Heath Road*
*Glossop*
*Derbyshire*
*SK13 9AY*

**OUR pleasure, David. As you can see, we are repeating repeating you again in a manner of speaking, with alternative versions of the 'correction'. The letter was one of those deliberate mistakes to which the letters page is peculiarly prone — but I hope it means that everybody got the message.**

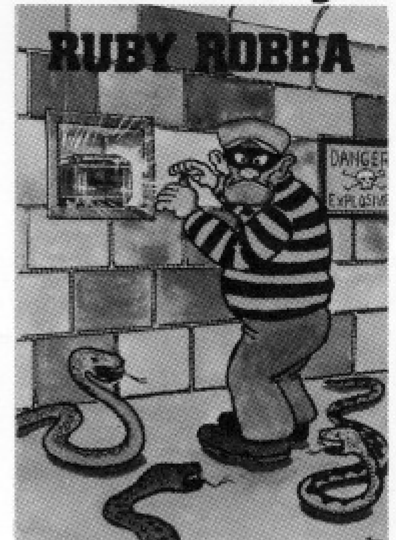
# News desk

If you have any new products for the Dragon — software or hardware — ring the News Desk on 01-437 4343

## Hey Preston — Blaby's back

THE stock of Dragon games formerly supplied by Blaby Computer Games has been taken over by Dragon country wholesalers R & A J Preston, who now have their own catalogue and price list ready for mail order.

As a wholesaler, Preston's handle software for a wide range of computers, and hope to use their wholesaling network to get more Dragon games back into the shops. Said Alan Preston, "We get letters from all over the place from people saying they can't get Dragon software locally. We actually had a call from a user group in Barcelona, and were able to send them a catalogue."




"We are also looking for new games and new writers. We would like to see the range increase considerably, and also obtain games which have only been available at full price up to now and put them out at budget price."

They are also supplying Dragon power packs, printer leads, Speedking joysticks, discs and other vital accessories.

Preston's free catalogue is available from the company at Kings Hall Court, St. Bridges Major, Mid Glamorgan CF32 0SE. Their 24 hour telephone number is 0656 880965, and orders can be placed on Access and Visa cards.

## New Gerrard

Peter Gerrard, brother of *Adventure Trail* writer Mike Gerrard, is taking over the column from this month.

Peter has particular experience in writing adventure games, and is starting a new series on this subject from this month. Peter has several books on computing to his name, not least *Exploring Adventures on the Dragon* (Duckworths) which may well be of interest to DU adventurers.

In fact, there'll be copies of the *tape* up for grabs in this month's Gordon Lee comp, for obsessive freebie-chasers. Over to you, Peter.

## Analyzer update

With reference to the corrections supplied by Mr. Pilz in the December edition of DU, we have an alternative update which may make the changes clearer to some readers. These follow as supplied:

```
11 POKES1,T+0:Z$
=STR$(S):IFS>.TH
ENZ$=RIGHT$(Z$,L
EN(Z$)-1):GOTO1E
LSE1
12 IFR>Z THENRET
URNELSEMID$(T$(X
),T+1,2)=RIGHT$(
STR$(Z+1),2):RET
URN
85 Z=VAL(MID$(T$
(X),T+1,2)):IFK=
.GOSUB12:NEXT:RE
TURN
114 IFA$(M)=""AN
DB$(M)=""ANDC$(M
)=""ANDD$(M)=""A
NDE$(M)=""ANDF$(
M)=""ANDG$(M)=""
ANDH$(M)=""ANDI$
(M)=""ANDJ$(M)="
"THENM=M-1:GOTO1
14
122 IFL(X)>=R TH
ENL(X)=L(X)+1
123 NEXT:FORX=7Ø
TOY:IFM(X)>R*Q T
HENM(X)=M(X)+Q
124 NEXT:GOTO29

DEL125
```

L.Pilz Australia

We also have this further communique:

"I CAME across another bug in *Analyzer* which gave me instant sweats and a rotten night. Could you please print these two lines ASAP:"

```
30 M=32:CLS:GOSUB50
31 K=1023:M=32:
POKEK+L,62:PRINT@447,""
```

It remains at present to say that in writing a long and complex program for *Dragon User*, Mr. Pilz has shouldered to the full an author's duty to straighten out any bugs. All the way from Australia.

## 6809 Show report

IT may have been work for many of the exhibitors, but I had a lovely time on Saturday 28th meeting *Dragon User* readers, advertisers and contributors (past and future, by the look of it) at the 6809 Show in London. Sorry, anyone I didn't get round to (and there were many) but I was due to turn into a pumpkin at two o'clock, so I had to leave smartish before that time.

We were well pleased with the write ups that we had from readers after the last show, so anyone who wants to send us a short report, please do so. Nice to see you, and thanks to everyone who attended. Support the Dragon, and the Dragon will support you.

## NDUG this month

The March edition of *Dragon Update* has just plopped through the door in time for the news page. This month we have an introduction to Flex (part 2), some advice on ASCII, mirror imaging screen dumps, copying autorun games, a review of an OS-9 comms program, COM09, written by Jason Shouler, some track listings from the original DeltaDOS user guide, many small ads, an exhortation from Paul Grade to get out and go to it? Shows, and a challenge from ditto to members to write to their regular computer magazines (*Dragon User* included), and say what they think of NDUG, good, bad and the rest. And he hopes I'll print these opinions.

Well, I'll print anything which I think is helpful to Dragon users, and indeed we have a piece on NDUG from Phil Beed this month, but I would like to toss in the suggestion that NDUG members write NOT, or at least not ONLY, to DU with their opinions, but to AS MANY other regular computer mags as they can think of. That is a way to reach Dragon users who are not yet in the safety net, the ones who will eventually write to us and say, 'We've never heard of you', despite our efforts.

Editors of other magazines may well feel uneasy about publicising *Dragon User* as coming from a rival stable, but they will not feel so constrained about giving NDUG a mention, because, as a user group, it has no commercial connections.

And why not write to the *Daily Mirror* and the *Daily Express* and the rest while you're about it?

## Another lost game

Nasty news. We've had evidence that a number of copies of *Wizard's Quest*, sent out as prizes for the November competition, didn't leave this building in the hands in which they should have left this building. Please will any prizewinner from that month who didn't get their *Quest* please drop us a line so that we can take steps to find out what happened.

**NOTE FOR NEW CONTRIBUTORS:** payment for published material will normally be received four to six weeks AFTER publication of the issue in question. All published features (a feature in DU's case is virtually anything which isn't a letter or a publicity release) and reviews are paid. News items are not paid unless they involve reportage by prior arrangement.

If the normal time has elapsed and you have not received an expected payment, please get in touch with Anne Marie who will be able to tell you when the mailing took place.

# SIMON

### *Iolo ap Gwyn makes a game of PCOPY*

I HAVE seen a number of versions of this game, none of which compare with the 'instant' response of the specially designed electronic game which has proved to be so popular. In order to be an improvement on these a program would have to give rapid responses, clearly written instructions, and while being difficult enough to present a challenge to an adult also be simple enough for a young child. The version presented here has proved very popular with my five, eight and ten year old children.

The key to this program is the use of the PCOPY command. By filling all the graphics pages with ready prepared instructions and coloured screens with corresponding numbers as well as keeping page 8 free as a dummy transfer page, what emerges is a game which instantly presents instructions in large clear letters without having to wait for them to be drawn each time. This particular command in Dragon Basic is very useful whenever any section of a graphic display needs to be either removed or introduced at some other time in the program. A description of the program will serve to illustrate the use made of PCOPY as well as the DRAW command in this game.

POKEing this location in line 20 will speed up the setting up of all the displays on machines that can tolerate its use, the program will run perfectly well without it. Lines 30 to 460 provide all the characters needed for the instructions given in this game. The REMs at the end of each line indicate which character is drawn with each string. Subroutines at lines 480 to 510 and 520 to 570 draw characters and strings at required locations on the graphics pages respectively. A subroutine at lines 610 to 640 draws solid borders around graphics pages while lines 650 to 670 are merely called when a delay is required, such as at the end of a game.

Setting up the displays commences in line 680, and a 'WAIT' instruction is included so that the player does not think there is anything wrong because nothing seems to be happening for a while. Lines 720 to 790 set up pages 1 to 4 of graphics each one with the corresponding DRAGON colour set colour as background and the number corresponding to that colour written boldly across it. The colours used are the primary red, green, blue and the yellow of colour set 0. These are the pages which are called when the game is running. Lines 800 to 1020 are taken up in writing all the instruction messages which are placed in graphics pages 5 to 7. Use is made of the character set defined at the begining of the program in conjunction with the character and line drawing subroutines at lines 480 and 520 to do this. The POKE command in line 1030 slows the CPU back to its normal rate of operation for running the game.

The sequence of colours/numbers/notes generated is held in a string called SQ$

```
10 CLEAR200:PCLEAR8
20 POKE&HFFD7,0
30 DIM A$(90)
40 A$(0)="BR7":'SPACE
50 A$(1)="BR2R0BU2U4":'!
60 A$(6)="BR1R0R1E2BD2D0H3E2H1L1G1F1D1G1D1F1":'&
70 A$(14)="BR2R0U1R1D1L1":'.
80 A$(16)="BU1U0U4E1R2F1D4G1L2H1F1U1E1U2E1U1":'0
90 A$(17)="BR1R0R2L1U6G1":'1
100 A$(18)="R0R4L4E4U1H1L2G1":'2
110 A$(19)="BU1U0F1R2E1U1H2E2L4":'3
120 A$(20)="BR3R0U6G3D1R4":'4
130 A$(21)="BU1U0F1R2E1U2H1L3U2R4":'5
140 A$(22)="BR1R0R2E1U1H1L3E3G3D2F1":'6
150 A$(23)="U0U1E2L1R2L1E2U1L4":'7
160 A$(24)="BR1R0R2E1U1H1L2H1U1E1R2F1D1G1L2G1D1F1":'8
170 A$(25)="BR1R0E3U2H1L2G1D1F1R3":'9
180 A$(26)="BR1R0R1U1L1D1BU3U0U1R1D1L1":':
190 A$(29)="BU2U0R4BU2U0L4":'=
200 A$(33)="U0U4E2F2D4U2L4":'A
210 A$(35)="BR1R0R2E1BU4U0H1L2G1D4F1":'C
220 A$(36)="R0R3E1U4H1L3R1D6":'D
230 A$(37)="BR4R0L4U3R3L3U3R4":'E
240 A$(41)="R0R4L2U6L2R4":'I
250 A$(44)="BR4R0L4U6":'L
260 A$(46)="U0U6F2D2F2U6":'N
270 A$(47)="BR1R0R2E1U4H1L2G1D4F1":'O
280 A$(50)="U0U6R3F1D1G1L3R1F3":'R
290 A$(51)="BU1U0F1R2E1U1H1L2H1U1E1R2F1":'S
300 A$(52)="BR2R0U6L2R4":'T
310 A$(57)="BR2R0U3H2U1D1F2E2U1":'Y
320 A$(65)="BU4BR1R0R2D3G1L1H1E1R1F2":'a
330 A$(67)="BR3R0L2H1U2E1R2":'c
340 A$(69)="BR3R0L2H1U2E1R2F1D1L4":'e
350 A$(71)="BD2D0R3U6L2G1D2F1R2":'g
360 A$(73)="BR1R0R2L1U4L1BU2BR1R0":'i
370 A$(75)="U0U6D4R1E2G2F2":'k
380 A$(77)="U0U3E1R1D4U4R1F1D3":'m
390 A$(78)="U0U4R2F1D3":'n
400 A$(79)="BR1R0R2E1U2H1L2G1D2F1":'o
410 A$(80)="R0R2E1U2H1L2D6":'p
420 A$(82)="U0U4F1E1R1":'r
430 A$(83)="R0R3E1H1L2H1E1R2":'s
440 A$(84)="BR3R0L1H1U5D2L1R3":'t
450 A$(87)="BU4U0D3F1E1U1D1F1E1U3":'w
460 A$(89)="BD2D0E3U1E1U1BL4L0D1F1D1F1":'y
470 GOTO680
480 'PRINT CHAR
490 X$=STR$(X):Y$=STR$(Y):S$=STR$(S)
500 DRAW"BM"+X$+","+Y$+"S"+S$+N$
510 RETURN
520 'PRINT LINE
530 FOR I=1 TO LEN(LI$)
540 D$=A$(ASC(MID$(LI$,I,1))-32)
550 GOSUB580
560 X=X+XS
570 NEXT:RETURN
580 X$=STR$(X):Y$=STR$(Y):S$=STR$(S)
590 DRAW"BM"+X$+","+Y$+"S"+S$+D$
600 RETURN
```

which is added to in lines 1070 to 1100 each time the player successfully enters the correct sequence of numbers. It is necessary to take the second character in the string in line 1090 because every number generated by the RND command is preceeded by a space when this number is converted to its string representation by the STR$ command. The loop between lines 1110 and 1160 copies the appropriate page to the top of the screen and produces the corresponding note. The length of time for which the number appears is controlled by the length of the note which gets progressively shorter as the player manages to remember longer sequences. The shuffling around of pages in lines 1170 and 1180 is necessary in order to avoid loosing any instructions and to ensure that the correct pages appear the next time around. This shuffling occurs at the other places as well for the same reason.

The response of the player is checked in lines 1190 to 1280. A correct answer produces a suitable display on the text screen. Since it would be a pity to return to the ordinary text mode at this stage, and since also there are no spare graphics pages available, then a different approach was considered desirable. The display is generated by POKEing in line 1330 the values contained in the DATA statements held on lines 1350 to 1490. The figures come in pairs, the first being the location and the second being the code for a graphics character. Lines 1510 to 1530 reset all pages and values and send us back to 1030 for another value to be added to the sequence and the display to run again, this time a little faster.

Incorrect answers are dealt with in lines 1540 to 1660, giving the number of correct answers in that particular set. The time taken to write out the response 'on screen' illustrates how important the PCOPY command is to the running of this game. Finally should any player reach the grand total of a sequence of 20 correct answers then the screen goes mad until filled and then gives the champion a chance to repeat the feat!

Should anyone not fancy the task of typing in all this program then I would only be too pleased to send them a casette of it if they send me four pounds (I. ap Gwynn, Plas Treflys, Llangwyryfon, Aberystwyth, Dyfed, SY23 4HD, WALES).

```
610 'DRAW BORDER
620 LINE(I,I)-(255-I,95-I),PSET,B
630 LINE(I,I+95)-(255-I,191-I),PSET,B
640 RETURN
650 'DELAY
660 FORI=0TO2000:NEXT
670 RETURN
680 SQ$="":AN$=""
690 CLS:PRINT@8+(5*32),"simon"
700 PRINT:PRINT:PRINT">>>>setting
    up>>>wait!"
710 PRINT@15*32,"c I.ap gwynn,9/83"
720 J=4:FOR I=1TO4
730 PMODE1,I
740 PCLSI:COLORJ
750 ST$=STR$(I):ST$=MID$(ST$,2,1):
    N=ASC(ST$)-32
760 N$=A$(N)
770 X=100:Y=80:S=32:GOSUB480
780 J=J-1
790 NEXT
800 'PREP INSTR
810 PMODE1,5:PCLS3:COLOR2,3
820 FOR I=0TO10
830 GOSUB610
840 NEXT
850 LI$="READY"
860 X=20:Y=70:S=32:XS=40:YS=40
870 GOSUB520
880 LI$="Simon says...."
890 X=30:Y=130:S=8:XS=12:YS=20
900 GOSUB520
910 LI$="Look & Listen!"
920 X=30:Y=160
930 GOSUB520
940 PMODE1,7:PCLS3:COLOR2,3
950 FORI=0TO10
960 GOSUB610
970 NEXT
980 LI$="Simon says..."
990 X=30:Y=40
1000 GOSUB520
1010 LI$="repeat now..:"
1020 X=30:Y=70:GOSUB520
1030 POKE&HFFD6,0:'RUN GAME
1040 PMODE1,5:SCREEN1,0
1050 GOSUB650
1060 PCOPY5TO8
1070 RN=RND(4):RN=INT(RN)
1080 RN$=STR$(RN)
1090 RN$=MID$(RN$,2,1)
1100 SQ$=SQ$+RN$
1110 FORI=1TOLEN(SQ$)
1120 P$=MID$(SQ$,I,1):P=VAL(P$)
1130 PCOPY P TO5
1140 SO=P*50
1150 SOUND  SO,20-LEN(SQ$)
1160 NEXT
1170 PCOPY8TO5:PCOPY6TO8
1180 PCOPY7TO6:PCOPY8TO7
1190 PMODE1,5:SCREEN1,0
1200 PCOPY5TO8
1210 FORI=1TOLEN(SQ$)
1220 IN$=INKEY$
1230 IF IN$="1" OR IN$="2" OR IN$="3"
     OR IN$="4" THEN1240 ELSE 1220
1240 PCOPYVAL(IN$)TO5
1250 SOUND VAL(IN$)*50,20-LEN(SQ$)
1260 AN$=AN$+IN$
1270 NEXT
1280 IFAN$=SQ$THEN1290ELSE1540
1290 IF LEN(AN$)=20THEN1670ELSESCREE
     N0,1:CLS0:RESTORE
1300 READN
1310 FORI=1TON
1320 READN1,N2
1330 POKE1024+N1,N2
1340 NEXT
1350 DATA 130
1360 DATA 33,129,34,140,35,137,52,138,55,133,65,133,68,129,69,140,70,137,71,132
1370 DATA 72,134,73,137,74,132,75,134,76,137,77,129,78,140,79,137,80,129,81,140
1380 DATA 82,137,83,132,84,142,85,136,87,133,97,133,99,129,100,133,102,133,104,1
38
1390 DATA 107,138,109,133,110,140,111,136,112,133,114,129,116,138,117,129,119,13
3
1400 DATA 130,140,131,136,133,140,134,136,136,136,139,136,142,140,143,136,145,14
0,146,136
```

```
1410 DATA 148,132,149,136,151,132,183,132,197,131,198,130,207,129,228,133,230,13
2
1420 DATA 232,131,233,130,235,131,236,130,238,131,239,135,260,133,261,129,262,13
1,263,133
1430 DATA 265,133,266,133,268,133,269,133,271,133,292,132,293,131,294,135,295,13
2,296,131
1440 DATA 297,134,298,132,299,131,300,134,301,132,302,131,303,135,305,129,308,12
9,311,129
1450 DATA 388,149,389,146,391,149,393,150,394,156,395,156,396,146,397,149,400,14
9
1460 DATA 420,149,421,148,422,146,423,149,425,154,428,154,429,149,431,146,432,14
9
1470 DATA 452,149,454,148,455,151,457,154,460,154,461,149,462,150,463,148,464,15
1,466,147
1480 DATA 468,145,469,146,471,147,484,148,487,148,489,148,490,156,491,156
1490 DATA 493,148,496,148,498,154,500,148,501,152,503,154
1500 GOSUB650
1510 PCOPY8TO5:PCOPY7TO8
1520 PCOPY6TO7:PCOPY8TO6
1530 AN$="":IN$="":GOTO1030
1540 'INCORRECT ANSWER
1550 LINE(0,0)-(255,95),PRESET,BF
1560 LI$="INCORRECT!"
1570 X=40:Y=20:S=8:XS=16:YS=20
1580 GOSUB 520
1590 LI$="Start again!"
1600 X=40:Y=55:GOSUB520
1610 LI$="Score="+STR$(LEN(AN$)-1)
1620 X=40:Y=85:GOSUB520
1630 FORP=0TO20:SOUND10,1:SOUND20,1:NEXTP
1640 GOSUB650
1650 PCOPY8TO5:PCOPY7TO8:PCOPY6TO7:
PCOPY8TO6
1660 SQ$="":AN$="":IN$="":GOTO1030
1670 'COMPLETE SET
1680 SCREEN0,1:CLS0
1690 POKE&HFFD7,0
1700 FORX=1024TO1535
1710 POKEX,RND(255)
1720 SOUNDRND(255),1
1730 NEXTX
1740 POKE&HFFD6,0
1750 SQ$="":GOTO1510
```

# Dragonsoft

New software for review should be sent to *Dragon User*, 12-13 Little Newport Street, London WC2H 7PP.

## Thinking person's boulders . . .

**Program:** *Stone Raider II*
**Supplier:** Microdeal
**Price:** £5.95

THE instructions supplied with this game bear no resemblance to the correct instructions, which scroll along the bottom of the title screen. I hope Microdeal can clear up this slackness.

*Stone Raider II* is a similar game to Blaby's *Boulder Crash*. The aim of the game is to collect all of the gems in the 25 different caves. The number of gems on each screen ranges from one to 125, but the number of gems to collect does not reflect the difficulty of the screen. Once all the gems have been collected you have to find your way to the flashing exit.

You control your man using the right joystick. Generally each screen becomes slightly more difficult as you progress through the game, but some screens later on seem easier than earlier ones. The 25th screen is extremely difficult, and I am yet to conquer it!

To pick up a gem all you need to do is walk over it. There are many nasty monsters to stop you in your quest (*By walking all over you? — Ed.*), as well as boulders and slime.

Different monsters behave in different ways, some always preferring to move left, others moving right. When certain monsters are killed by boulders or by gems, they will turn into nine gems. On some screens there are more gems than you actually need to complete the screen. In order to trap the monsters you need to do some clever trail laying so that the monsters will follow your track to where you can drop boulders on them, which all goes to make the game far more enjoyable.

Falling boulders are another problem. Boulders fall when they are pushed, or when the ground is pulled away under them. The falling boulders will kill you if you don't move fast enough.

*Stone Raider II* is not the usual shoot'em up and knock'em down game. It requires skill, and it is usually necessary to think about your next move, as moving the wrong way, moving too soon or too late could either get you trapped in a rock fall or killed off altogether.

There are 25 screens, and considering you only start with three lives the writer has included a cheap facility which enables you to change the number of lives, start screen and speed bonus. This facility is an asset for people who are not too adept at games playing, and also enables each screen to become a game in its own right.

Obviously a lot of thought has gone into each screen to make it more difficult than it first appears. I would recommend this game to any arcade or even adventure fan who enjoys a bit of fun. *Stone Raider II* is the sort of game you can play and play until you have completed a screen, just to prove to yourself it can be done.

*Robert Margrave*



## DOS DISC ROM

**Program:** *DOSplus* 2.7 Disc ROM
**Supplier:** P. G. Scott, 4 Badgerwood Drive, Frimley, Camberley, Surrey GU16 5UF
**Price:** £15

OH NO! Not another DragonDOS compatible ROM? Well, yes, but this one has all the bugs removed! What do you get for your money? A little black thing about 2.5 inches . . . oh, you mean the program. Right. The DOS in this ROM gives all the facilities of DragonDOS, plus a number of DragonDOS command extensions, plus extensions to other Basic commands, plus other extras (that's why it's called DOSplus).

Firstly, DragonDOS compatibility. All the standard commands are there, though some do give different results if used in the 'standard' DragonDOS way (your programs are unaffected). These include DIR scrolling slowly to the screen (you can just about read it), with a disc name at the start and a file count at the end; DSKINIT checking that you really want to format the disc in the drive and displaying the track being processing; BACKUP on a single drive stopping the motor quickly; EOF performing the same with cassette and disc. There are also extra error checks — it is now difficult to make your drive sound like a machine gun when the limit stop is reached!

A number of DragonDOS commands have been extended to give extra facilities: DSKINIT will write a disc name when formatting, RENAME will rename discs, DIR can be made to page directory lisitings and output to any open output stream (ie printer, cassette and disc file), EOF, LOC, and LOF will accept filenames or stream numbers (just be patient, I'll get to that later), LOC can also be used to set the read pointer, LOAD will load segmented binary files and KILL can be used with a special filename to work through a directory asking for files to be deleted.

However, in addition, the following enhancements are thrown in free. RESTORE will accept a line number, OPEN can be used to open disc files in four modes (input, output, append, empty), allocating a steam number in the range 1 to 10. (If you are not sure what I mean, the cassette is stream -1 and printer is steam -2). To complement this, INPUT, LINE INPUT and PRINT have been enhanced to work with disc streams and CLOSE has been extended as well.

Finally, you can select different keyboard modes (rollover or repeat) and use the standard character input/output routines for disc I/O, from machine code, in the same way as for cassette (but easier).

That's the lot! and it all fits into the same size ROM as DragonDOS. Now FLEX 1.3.1 does not boot, but Phil Scott will supply the FLEX corrections with the ROM on request and the DOS is fully supported, with all problems investigated and your EPROM reprogrammed with the corrections for 50p.

Give yourself a treat and get DOSplus 2.7, the ultimate update of the ultimate DOS for the Dragon.

*Robin Warwick*



## Tracks per pound

**Program:** *Diskfix.*
**Supplier:** H. C. Andersen Computer a/s, Englandsvej 380, DK-2770 Kastrup.
**Price:** £80.00 plus p&p.

A COUPLE of months ago I reviewed one of H. C. Andersen's new utilities. The second is *Diskfix*.

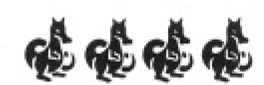
This is not, as it may first sound, yet another disc doctor program. In fact it provides support for 80 track drives to be used with your standard version of OS-9. When run the program will allow you to select one or two sided discs, 40 or 80 tracks. It then formats the disc to your specification and writes new 80 track drivers to the disc (it creates a new version of OS-9 boot for those that want to know).

If 40 tracks are selected then double pulses are sent to the drive for track movement — ie the 80 track drive moves two of its 'tracks' for every one of a 40 track disc. Of course, it goes without saying that you must have the hardware to support all of this, ie a single or double sided 80 track disc drive; you cannot use a 40 track disc drive (such as the standard Dragon Data drive) with this software.

Once the disc has been formatted and the drivers installed, the Diskfix program is not needed to use the 80 track discs. The advantage is that (with a suitable drive) you get four times as much space to one disc as you would using the standard Dragon drive and format program. The disadvantage is price — £80.

If you're an OS-9 user and have 80 track drives then you must decide for yourself on the value of this software; it certainly does exactly what it claims to do. If you don't already have a 80 track drive, then the cost of one plus this software would add up to an awful lot of blank discs.

*Brian Cadge*



Requires: Dragon 64 with disc drive and OS-9 system.

# Searching techniques

***Dene Bebbington** shows you how to find quickly what you're looking for*

MANY users have probably used some form of searching while programming as the need to search lists of data arises in many applications — eg databases, mailing lists etc. In these applications searching is often one of the most used processing activities, so it is desirable to keep the time spent searching to a minimum. Searching methods will vary greatly in performance.

When measuring the performance of searching routines we count the number of probes which are made in the list to find the item we are looking for. If we use N to represent the number of items in the list then we find that the number of probes to find an item can vary from N to just 1 depending on the method used. If we choose a good method, searching time can be greatly reduced.

The method chosen depends on whether the list is sorted or not. Most of the time it is much faster to search a sorted list than an unsorted one, but there is an exception in the case of hashing — this will be looked at separately.

## Unsorted lists

If we have an unsorted list to search there is a simple technique we can use. Start at the beginning of the list and examine each successive item until the one you are looking for is found, or until you reach the end of the list if it is not there.

This method is known as a LINEAR SEARCH and is easy to implement. See **listing one** for an example. Here K$ is the item being searched for and the list is held in the array A$() of length N. If K$ is not in the list then control is transferred to line 50, otherwise the program jumps to line 70 with its position stored in I.

**Listing 1**

```
10 I=0
20 I=I+1
30 IF K$=A$(I) GOTO 70
40 IF I<N GOTO 20
50 PRINT"NOT FOUND"
60 END
70 PRINT"FOUND AT ";I
80 END
```

This method is not suitable for anything but small lists as in the worst case (where the item is at the end of the list) it needs to make N probes. In the best case (where the item is at the start of the list) it will only need to make one probe. Thus, on average, it will only need to make N/2 probes for successful searches. For unsuccessful searches it will always have to make N probes.

## Sorted lists

If we have a sorted list then we can make use of its order to improve searching times. We could use the ordinary linear search but this would be inefficient as it does not take into account the ordering of the list, and would make unnecessary probes.

Firstly there is the 'intelligent' linear search, which is essentially the same as the ordinary linear search except that we can stop searching if the item we are looking for is less than the current item being looked at (see **listing two**). Below is an example. The item we are looking for is the name Harry, and here we would stop searching at the name Henry.

**Listing 2**

```
10 I=0
20 I=I+1
30 IF K$=A$(I) GOTO 70
40 IF K$<A$(I) AND I<N
   GOTO 20
50 PRINT"NOT FOUND"
60 END
70 PRINT"FOUND AT";I
80 END
```

LIST: Alan David Eric Gordon **Henry** Ian Mark Paul Robert

This search method only has to make N/2 probes on average whether the item is in the list or not, which makes it better than the ordinary linear search for sorted lists.

Secondly there is the BINARY SEARCH which is a very efficient searching method for any size of list. The way it works is to split the list in half and see if the middle item is the one we are looking for; if not, it works out which half it should be in and then repeats the process.

Thus we are halving the size of the list to be looked at after each probe. Below is an example of the probes we would need to make to find the name Paul.

| **List** | (1) | (2) | (3) |
|---|---|---|---|
| Alan | | | |
| David | | | |
| Eric | | | |
| Gordon | | | |
| Henry | Henry | | |
| Ian | | | |
| Mark | | Mark | |
| Paul | | | Paul |
| Robert | | | |

If an item is not in the list then searching is stopped when the length of the sublist being examined is 1.

Using the binary search we find that if an item is in the list then only 1 probe is needed in the best case, and only log N (base 2) probes are needed in the worst case. If the item is not in the list then log N (base 2) probes will always be needed. We can see that the binary search is significantly better than the other two methods looked at. It is not very difficult to code and is thus recommended in applications where sorted lists are used (see **listing three**).

**Listing 3**

```
10 B=1
20 T=N
30 M=FIX((B+T)/2)
40 IF K$=A$(M) GOTO 90
50 IF K$<A$(M) THEN
   T=M-1 ELSE B=M+1
60 IF T>=B GOTO 30
70 PRINT"NOT FOUND"
80 END
90 PRINT"FOUND AT ";M
100 END
```

## Hashing

Another form of searching is hashing, which is not strictly a form of searching but a way of inserting items into a list, leading to efficient searching. A hashing function is applied to the item which generates the position in the list where it is to be inserted. A problem arises when a collision occurs (where the position is already occupied). Some way of handling the collision is needed.

If we can derive a hashing function which results in a low number of collisions then searching will be faster. If we could find a hashing function which leads to no collisions then all searches would only need 1 probe, however, we can't really expect this and so searching using hashing will take more than 1 probe but will tend to be more efficient than the other methods looked at.

Generally we find that the best results are obtained when the size of the list is a prime number, for example if we want to hold about 100 items then it is best to use a list of size 101 which is prime.

Here I will present a hashing routine which leads to quite good search times, on average. A secondary hash function is used if a collision occurs and if a collision occurs

again then the item is inserted in the next free position in the list — this is known as LINEAR PROBING.

The way it works is to add up all the ASCII values of the characters in the item (K$) and then to multiply this by the length of the item, giving T. The value of T mod N is then calculated and 1 is added, this is then used as the position in the list for insertion. If there is a collision then we use the secondary hash function which is the original hash function plus the length of the item. If there is also a collision at this position then we examine each consecutive location until a free position is found, whilst doing this we 'circle' to the beginning of the list if the end is reached.

**Listing four** is an implementation of the method set out here, and it could easily be converted for use in a database, stock control system etc., while the hashing function could be changed to improve the performance.

As can be seen from the program searching is done by applying the hash function to the item being searched for. If the item is not at this first location then we apply the secondary hash function to the item, if at this next location the item is not found then linear probing is used until it is found, or until we circle back to the first location examined.

Another searching method is available to us but can only be used on sorted lists. I call this the PARTITIONED LIST SEARCH as the way it works is to partition the list so that only the relevant partition need be searched. For example, if we are dealing with items consisting of just letters of the alphabet then we can partition the list into 26 parts, each part containing items beginning with a different letter. Thus the first partition contains all items beginning with 'A', the next with 'B' and so on.

We can use an array as a table to hold the partition information, the first entry in the table says where items beginning with 'A' start. The next where items beginning with 'B' start, etc. When searching we can then use this table to find the relevant partition to search.

See below for an example list and table.

| LIST | PARTITION TABLE | |
|---|---|---|
| A 1 | 1: 1 | 14: 9 |
| A 2 | 2: 3 | 15: 0 |
| B 3 | 3: 4 | 16: 11 |
| C 4 | 4: 0 | 17: 0 |
| C 5 | 5: 0 | 18: 0 |
| F 6 | 6: 6 | 19: 13 |
| G 7 | 7: 7 | 20: 0 |
| G 8 | 8: 0 | 21: 16 |
| N 9 | 9: 0 | 22: 0 |
| N 10 | 10: 0 | 23: 18 |
| P 11 | 11: 0 | 24: 0 |
| P 12 | 12: 0 | 25: 0 |
| S 13 | 13: 0 | 26: 19 |
| S 14 | | |
| S 15 | | |
| U 16 | | |
| U 17 | | |
| W 18 | | |
| Z 19 | | |
| Z 20 | | |

As can be seen from example, all items beginning with 'A' start at position 1, those beginning with 'B' start at position 3 and so on, ending with those beginning with 'Z' starting at position 19. Any entries in the table with zero mean that there are no items beginning with the correspondents letter. Also the next non-zero entry in the table represents the end of a partition.

To implement this search we need 2 routines, the first routine creates the partition table **(listing five)** and the second routine finds the end of the partition which the item being searched for is in **(listing six)**.

Using this method we can apply either the linear or binary search to the appropriate partition, the improvement in performance is greater with the linear search than with the binary search. The improvement comes from the fact that it only needs to examine likely items, for example, if we are searching for the name Paul then we need only look at the items beginning with 'P'.

The searching method chosen for use in an application will depend on whether sorted lists are being used, what performance is required, and also how much space is available. Generally the binary search is a good choice for sorted lists, and hashing for unsorted lists.

**Listing 4**

```
10 ' MAIN PROGRAM WOULD GO HERE.
20 B$=STRING$(255," ")
30 DEF FN H(X)=X-(FIX(X/N)*N)+1
100 ' ROUTINE FOR INSERTING ITEMS.
110 ' ASSUMES DIM A$(N) HAS BEEN DONE.
120 T=0
130 FOR I=1 TO LEN(K$)
140 T=T+ASC(MID$(K$,I,1))
150 NEXT I
160 T=T*LEN(K$)
170 IF A$(FN H(T))<>B$ GOTO 210
180 A$(FN H(T))=K$
190 RETURN
200 ' SECONDARY HASH
210 NP=FN H(T)+LEN(K$)
220 IF NP>N THEN NP=NP-N
230 IF A$(NP)<>B$ GOTO 270
240 A$(NP)=K$
250 RETURN
260 ' LINEAR PROBING
270 NP=NP+1
280 IF NP>N THEN NP=NP-N
290 IF A$(NP)=B$ THEN A$(NP)=K$ ELSE GOTO 270
300 RETURN
```

```
200 ' ROUTINE FOR SEARCHING.
210 ' L$ IS ITEM TO BE FOUND.
220 ' F IS POSITION L$ FOUND AT.
230 ' IF L$ IS NOT FOUND THEN F=0.
240 F=0
250 T=0
260 FOR I=1 TO LEN(L$)
270 T=T+ASC(MID$(L$,I,1))
280 NEXT I
290 T=T*LEN(L$)
300 IF A$(FN H(T))=L$ THEN
    F=FN H(T):RETURN
310 NP=FN H(T)+LEN(L$)
320 IF NP>N THEN NP=NP-N
330 IF A$(NP)=L$ THEN F=NP:RETURN
340 NP=NP+1
350 IF NP<>FN H(T) GOTO 320
360 RETURN
```

**Listing 5**

```
100 ' ASSUMES P=26.
110 ' ASSUMES DIM T(P) HAS BEEN DONE.
120 ' ARRAY T IS THE PARTITION TABLE.
130 I=0
140 I=I+1
150 C$=LEFT$(A$(I),1)
160 X=ASC(C$)-64
170 IF T(X)=0 THEN T(X)=I
180 IF C$<>"Z" AND I<N GOTO 140
190 RETURN
```

**Listing 6**

```
200 ' L$ IS ITEM BEING SEARCHED FOR.
210 ' E IS THE END OF THE PARTITION.
220 C$=LEFT$(L$,1)
230 I=T(ASC(C$)-64)+1
240 IF T(I)<>0 THEN E=T(I):RETURN
250 I=I+1
260 IF I<P GOTO 240
270 E=N
280 RETURN
```

# Introduction to wordprocessing

***Roger Merrick** checks off the vital points of computerised text handling in the first of two articles*

IT seems to me that for anyone with a need to do any written work on paper, and preferably some experience with a manual typewriter, the single most useful purpose that a home computer system can easily serve is that of word processing.

For the beginner, or the unconvinced, the computer must be able to demonstrate some advantages over the traditional methods of pen — or typewriter — and paper. Why would I use a word processor when I could scrawl something on a piece of paper with a biro?

Given that the hardware and software requirements are met, a wordprocessing system offers the following advantages over the alternatives:

a) high standard of presentation — readable, neat, nothing crossed out, no Tippex.

b) easy reworking of text — if you've typed or written a page, you feel less than keen to rewrite it if a better way of expressing yourself occurs to you (or worse, to someone peering over your shoulder). On the wordprocessor, it's a piece of cake.

c) work can go from rough notes to completed text in one operation. Because it is easy to reformat text, I tend to put my original thoughts straight into the computer in the order that they occur to me. Then I can expand, reorganise, present, edit, as I wish.

d) repetitive typing can be ended — instead of typing 'From: A. Bloggis, 21 Railway Mews' etc. at the start of each letter, you can load it from disc with a couple of keystrokes; standard letters can be stored and customised as required.

e) a word processor can be used as a database, archiving books, articles or records, names and addresses, as a diary, and so on.

For the computer system to stand any chance of competing with the biro certain minimum requirements must be met.

The computer system consists of three important components:

1) the computer system hardware
2) the wordprocessing software
3) the user.

First, to consider the user:

a) the user must be committed to using the new system because there will be a latency period during which it will be slower to use the computer than the alternative

b) the user must have a need to be met by the wordprocessing system. I mean that if your writing needs per annum are half a dozen thank you letters to relatives and some apologies to the Bank Manager, then it is hard to justify the cost of the word processor, and it is hard to generate the time using it necessary to learn to use it efficiently: whereas if the user is a student having to prepare and submit substantial amounts of written material, or uses a word-processor at work, or has any special interest that involved generating written work, then, given that other conditions are met, a word processor will offer a significant improvement over alternative methods of generating text

c) the user needs an ability to learn, to follow instructions, to accept his or her own lack of understanding rather than always blaming the equipment, yet to infer when the instructions provided are incorrect and

d) ideally, will have already got past the very worst 'hunt and peck' stage of typewriter keyboard literacy. You don't need to be a touch typist to use a word processor; indeed, in my opinion it is undesirable (because touch typists are taught to type in a certain way because of the constraints of the mechanical keyboard), but you would find it helpful to have a pretty good idea of where each key is.

Secondly, the hardware:

a) unless the hardware is a self contained portable system (in which case this point goes without saying), the computer system should be always in a condition where all the user needs to do is switch it on (and, I will accept, load the software) — you're on to a loser if, instead of picking up a biro, you've got to beg the TV off the kids or the loved one, carry it to the computer room, find a three way plug adaptor etc., etc., messing about for an hour

b) the system should consist of:
computer, monitor, printer, one or more disc drives.

The computer should have:

i) upper and lower case,
ii) at least 40 by 24 character display, more is better,
iii) near typewriter quality keys,
iv) standard qwerty key layout (including punctuation keys, shifted numbers (!'£$%&') and so forth)
v) memory of a size sufficient to allow, after operating system and wordprocessing software, about six A4 pages of text. More free RAM is fine, but not less. You see, I have this idea that there is some statistic to say that a typical document can be contained in well under this amount of space (*Don't you believe it. The shortest letter effectively ties up 1 or 2K, and anyone doing stories or essays will go bananas without a disc drive — Ed.*) but it is undesirable to be always filling up or running out of memory space, which adds to the time required, saving a document, breaking it into a number of parts, and slows down the operation of the software as, for example, string clearing operations have to be carried out increasingly often as the memory fills. The wordprocessor I use at work has about 37K free at maximum, and I have never run out of space. Though I write on it every day, most documents I create are far less than six pages in length.

The monitor should be high quality, high resolution, because the user will be spending quite long periods of time up close to it. The screen should be large enough to display characters that the user can read without strain. Black and white monitors are relatively cheap (£50 to £100). It should be positioned at an appropriate height so that the user does not need to crane his/her neck to look at it, and so forth.

Word processing does not require colour or sound, but whether either of these are desirable will depend on the software. Colour monitors of sufficent quality still cost a lot of money. Amstrad's colour monitors are not, in my opinion, of high enough resolution to be really usable in wordprocessing unless the working situation permits the screen to be positioned at some distance from the user.

(*Speaking as someone who spends whole days at w/p monitors, including a standard Amstrad PCW monitor, as well as cheaper and more expensive ones, I would recommend that anyone using any monitor sits at least two feet and preferably three feet away from it, as the glare from the screen is tiring to the eyes even when the resolution is first class. Experiment till you find a comfortable position. — Ed.*) I must acknowledge that I have seen very good results obtained on a 14" black and white TV and a Dragon running *Telewriter.*

The printer should use ordinary paper:

- not thermal (expensive, not always easy to get hold of, poor quality compared to typing paper)
- not narrow width because it limits the uses to which the output can be put

The printer should use an ordinary ribbon or a standard cartridge

- not strike once cartridges or little spirit pens because they are too expensive in use
- not highly specialised ink units, eg the solid ink in a glass tube of the Olivetti Sparkjet printer, because you try getting them when the Sparkjet is discontinued (dealers will, of course, offer 'continued support while there is a demand', like Boots did for the Dragon)
- which conforms to a current standard — eg 8 bit, Centronics plug if parallel
- which is able to produce the full ASCII range of characters
- which produces print that is of readable quality.

In my opinion, the speed that a printer works is not a major problem in word processing — you can always have a cup of coffee while the printer is working. (*Once again, this is fine for letters or invoices, but it can be a real problem if you are creating longer documents, or writing to a deadline — Ed.*) If you buy a standard machine, it may come with, or you may buy, a 2K or larger buffer if you need it. Alternatively, your software may support 'background printing', where printing is done without locking up the keyboard. The critical feature of a printer is the quality of the

print — if it's not satisfactory you can forget that printer. When I used a DMP110 I would always select a correspondence quality font even though it took a bit longer to print.

The different types of printer can be described in a nutshell

i) **Dot-matrix**, Epson or compatible. This is the effective current standard in dot-matrix printers. Fast, medium quality print, widely available ribbons. Some compatibles use ordinary typewriter ribbons that only cost about £ 1.50.

ii) **Daisywheel**. These produce the best quality print but are noisy. It is expected that developments in high quality dot-matrix printers will knock the bottom out of the daisywheel market, so a) don't spend a lot of money and b) make sure you get a standard one.

iii) **Penplotters.** For example the CGP115. A couple of years or so ago this was the new big thing. I found them unsuitable for the following reasons:

a) narrow, till-roll type paper

b) pens were of variable quality and short life — they would run out or go faint during a quite ordinary length document; at 50p each, they are expensive

c) the character set in the 8 inch version is very poor (it is good on the standard CGP115, however)

d) idiosyncratic operation — commands are unlike any other printing device.

Essentially, the machine is being used out of spec. — as a plotter it works great, as a printer it's limited.

iv) **Others:** laser printers work great but are too costly for home use; Sparkjet printers are around in limited numbers and you'll be stuck for supplies of ink when production is discontinued, some give poor quality lettering — the ink blurrs, the character sets are poor; thermal printers give highly variable quality print and thermal paper is limp and feels funny — your work will give a poor impression if presented in this way.

The data storage device should be one or more floppy disc drives. While a friend of mine who is a writer manages to do his work on a cassette-based system, it is obvious that the full advantages of the wordprocessing system will not come about until a disc is connected. The disc offers FAST saving and loading, encouraging regular backups — this article loads into *Telewriter* in a couple of seconds.

DragonDOS offers an automatic backup facility, discs are far more reliable than cassettes, under normal circumstances the documents saved are automatically catalogued (DIR) — you don't need to load a file before you discover what it is. It is practical to use a letterhead from disc — on cassette it may take longer to load a standard opening paragraph than to type it. The range of advantages available from using discs depends on how many drives you have and what the software you are using offers (Mailmerging or Spellchecking, for examples).

The alternatives in mass storage are worth mentioning:

i) 5.25, 3.5 and 3 inch, single or double sided, 40 or 80 track, half or full height, floppy disc drives are all functionally equivalent on the Dragon with DragonDOS, These are the only valid choices for mass storage.

ii) cassette: the Dragon's COS is not the slowest, not the least reliable, but it does not compare with the speed and reliability of the disc system.

iii) 8 inch drives can, I understand, be connected to the Dragon DOS cartridge. I haven't tried it. 8 inch drives and discs are disproportionately expensive compared to other forms of drive. If you can get one dead cheap, it might be interesting to play with.

iv) proprietory fast tape drives — eg Eurohard's Quickdrive, UltraDrive, Hobbit, Exatron Stringy Floppy etc. These are OK if they don't require special tape cartridges (ie use audio cassettes), but if they do, then when the model is superceded or the company goes bust, you're stuck. Also, the operating system may not be easily compatible with your mate's disc operating system or whatever; if you ever upgrade to another machine (God forbid!), your proprietary storage device is unlikely to be usable with the new system (you can just plug your disc drives into most systems — in my case, when I switched my Olivetti drives from a Video Genie to a Dragon, I literally just disconnected one cable and connected another); they are not as fast or as reliable as disc drives. No fast tape system can offer random access in the way that a floppy disc drive can.

v) hard disc system. These, which not so long ago would have been prohibitively expensive for a home computer system are now looking to be in the range of the well-off enthusiast, or the home businessperson. The snag is that they will require a special operating system and interface before they can be connected to the Dragon. You would need custom software too. An operating system like OS-9 would really come alive with a hard disc (and half a meg. of RAM).

There are no other mass storage devices of any relevance.

So how does the Dragon meet the hardware requirements?

It has a near typewriter quality keyboard, and in terms of the number of keys and their layout, closely matches a conventional typewriter.

It offers a standard parallel printer port, TV or composite video monitor output, easy disc interfacing.

In whatever configuration, 32, 'regular' 64, or upgraded 32 to 64, it offers a useful amount of memory.

As it stands, it does not give upper and lower case. There are facilities for software-generated upper and lower case drivers, but obviously they consume RAM. Have you ever wondered, by the way, why the VDG doesn't have upper and lower case? It seems it was to save a small bit of RAM. Remember that the original Tandy CoCo was offered with 4K or RAM, and the handful of bytes involved for upper and lower case made a difference then. If you have an expanded 32, it is possible to hide a high res screen driver in the upper 16K of RAM and still have disc I/O.

With the 51 by 24 screen, the Dragon offers more characters on screen than most other home computers (to quote from the *Telewriter* advertising).

If you are reading this, you are likely to have a Dragon. It connects via a cheap cable (£8 now, from Tandy) to a standard printer. You could get a secondhand Epson compatible printer for under £100, or not much more. If you don't have discs, the most expensive requirement is the DOS cartridge. Make sure you get the DragonDOS or CumanaDOS versions. PremierDOS, which once looked so good, is now a definite no no. New drives are costing £50 or so, with PSU and case extra, if you shop mail order. You'll need your own screen. The cost of gettting the show on the road would buy a hell of a lot of biros.

Finally, the software:

Wordprocessing software should meet certain general criteria:

i) it should be easy to use.

ii) it should prevent the user from doing anything really stupid, or, perhaps preferably, give the user a chance to change his/her mind before proceeding.

iii) it should not stand in the way of the user doing what s/he wants with the text.

iv) it should enable the user to access all relevant computer facilites from within the program.

v) it should take advantage of the facilities of the hardware to provide a fast-acting system.

vi) it seems self evident to me that the operating system and wordprocessing software together should leave an appropriate amount of memory free to use for text generation. In a disc based system, operations will be significantly slowed down if continual disc accesses have to be made.

vii) it should not attempt to be a jack of all trades — it should be designed to enable the user to generate text quickly and easily, not clog up memory and slow down its speed of operation with functions of limited, specialised, or no, relevance to text generation.

viii) it should store data (text) in a standard form so that files can be taken from one program to another for specialised applications.

ix) it should be able to be configured to the system being used.

For display, the user should be able to select foreground and background colours (limited only by hardware, not software), for storage, single or twin drives, 40 or 80 track, single or double sided, for printing, the printer should be driven by a user-configurable module, specifying for example, whether or not the program should send linefeeds with carriage returns, for typing, the keys on the keyboard should perform the functions engraved upon them. If the program has been written to run on a number of different machines, it should be possible, as with some games, for the user to select what keys perform what functions. The user configurations should be able to be stored so the user does not have to type them all in every time the program is used.

Essentially, you need to be able to match the 'up and away' convenience of winding a bit of paper into a typewriter and then, on top of that, gain the advantages of the computer system.

# Communication

Problem: Can anyone help? I have just got a Dragon and my son wants a copy of *World of Flight*.
**Name:** Bob R. Caffin
**Address:** 40, FD, REGT RA Training Wing, BFPO 30.

**Problem:** I need a machine code long division (base 10) routine. The divisor and dividend both to be poked, one digit at a time, into memory locations between 2000 and 3000 (decimal). The quotient need not be stored as I only require to know if the division is exact.
**Name:** Mark Towlson
**Address:** 19, Milldale Road, Long Eaton, Nottingham, NG10 3JB.

**Problem:** I am getting an out of memory error using my Dragon 32 software with CumanaDOS package in situ. Any ideas what is going wrong?
**Name:** D. S. Clarke
**Address:** 184, Favell Drive, Parkside, Furzton, Milton Keynes MK4 1AT.

**(Everybody wants to know about Cumana DOS and nobody knows the answers — come one, someone! — Ed.)**

**Problem:** I am interested in information about hardware add-ons for the Dragon 64. Is it possible to have more than 128K of memory on a Dragon? I would also like to swap programs, hints and tips etc. with other Dragon users. I own a Dragon 64 plus disc drive (DragonDOS) and printer.
**Name:** 7 Kent Vilhelmsen
**Address:** 7 Hydalveien 12, 3970 Langesund, Norway.

**Problem:** I would like to obtain an original copy of the game *Total Eclipse*. I would be willing to pay between £2 — £3 for the game. I would also want the instructions with the game. I also desperately need help with *Tanglewood*, and finally is there anybody out there who has a keyboard compatible with the Dragon?
**Name:** Paul McCormack
**Address:** 1 Mill Villa Clackmannan, Scotland FK10 4HJ.

**Problem:** Does anyone have the DragonDOS version of *Elite Calc*?
**Name:** C. Jobson
**Address:** 23, Walsingham, Biddick Village, Washington, Tyne & Wear, NE38 7HF.

**Problem:** Does anyone have a tape backup program which they would sell me?
**Name:** The Expert *(Not* **The** *Expert — Ed.)*
**Address:** 25 Fife Street, Dufftown, Leith, Banffshire, AB5 4AL.

**Problem:** I need some info on how to interface and build some simple "robotics" devices, eg arms etc. It also has to be very cheap. Also any routine for printing sprites in PMODE4 would be appreciated.
**Name:** Barry Caruth
**Address:** 132 Donaghadee Road, Bangor, Co. Down, N. Ireland.

Write down your problem on the coupon below (make it as brief and legible as possible) together with your name and address and send it to Communication, Dragon User, 12/13 Little Newport Street, London WC2H 7PP.

**Problem** ..............................................................

..............................................................

..............................................................

..............................................................

**Name** ..............................................................

**Address** ..............................................................

..............................................................

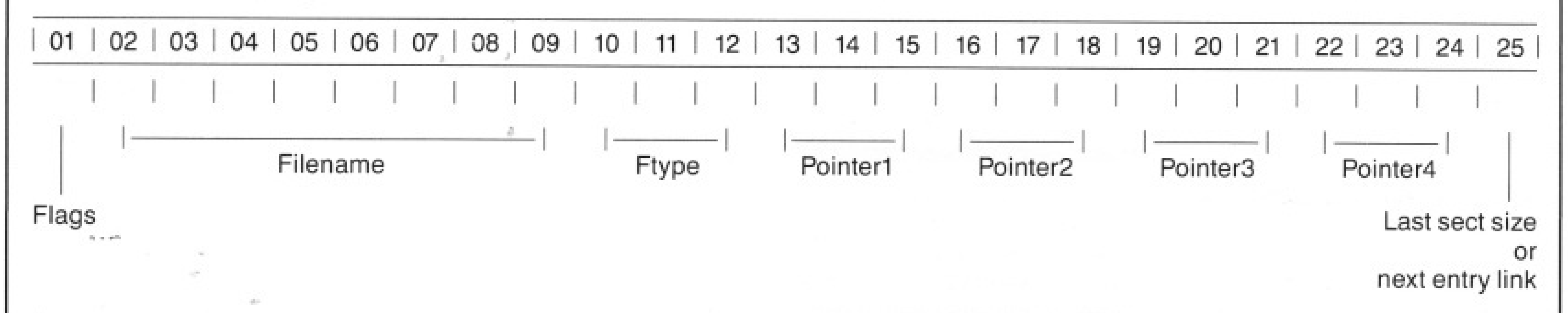
# Into the directory

## *Paul Dagleish opens up the Dragon DOS directory for programmers*

THIS article is intended to give an overall view of the organisation of the Dragon DOS directory. I was prompted into writing it when I read the article by D. Rothery on page 8 of *Dragon User*, November 1986. He did describe a lot of the information presented here but ignored parts which tend to cause difficulty to programmers trying to write 'directory access' programs for themselves; notably when directory entries have been extended by DOS. I hope this complementary information will help you to understand more clearly how his program works. I have repeated a fair amount of what he said, but this is because I have aimed at presenting a complete reference document.

As most regular readers of *Dragon User* now know the Dragon DOS directory is stored starting at the third sector of track twenty and continues to the end of the track, a total of 16 sectors. The whole directory is organised into 'slots' each of which is 25 bytes long. Thus each sector has room for 10 slots with 6 bytes left over at the end. (It is these free bytes which are used in Mr. Rothery's program to store a disc title.) This, then, provides us with a theoretical maximum of 160 files on a disc. Although each file has only one 'entry' in the directory, each entry can be composed of one or more 'slots', thus reducing the possible number of files in the directory. This distinction between 'slots' and 'entries' must be always born in mind.

The first slot of a directory entry always has the same format, as shown diagrammatically here:




Each 'initial' slot is divided into eight fields:

| | | |
|---|---|---|
| byte | 1 | : Flags |
| bytes | 2 to 9 | : Filename |
| bytes | 10 to 12 | : Filetype |
| bytes | 13 to 15 | : Pointer 1 |
| bytes | 16 to 18 | : Pointer 2 |
| bytes | 19 to 21 | : Pointer 3 |
| bytes | 22 to 24 | : Pointer 4 |
| byte | 25 | : SIZE/LINK byte |

These fields are described in detail below.

**Byte 1 entry status flags**

| Bit No. | Names | Meaning |
|---|---|---|
| 0 | Entry Type | 0 = Header entry (filename)<br>1 = Continuation (of another entry) |
| 1 | Protection Flag | 0 = Not protected<br>1 = Protected |
| 3 | Directory end flag | 0 = More entries in directory<br>1 = End of directory |
| 5 | Continued flag | 0 = No continuation<br>1 = This entry is continued |
| 7 | Valid file flag | 0 = This file OK<br>1 = File has been deleted |
| 2,4,6 | | Not used |

**Bytes 2 to 12, filename and filetype**

Bytes 2 to 9 contain the ASCII filename (eg 'THISPROG'), and bytes 10 to 12 contain the ASCII filetype (eg 'BAS'). If the filename is less than eight characters long then the zone for the filename in the directory is padded out with zeros. The filetype zone is padded to three characters in a similar way. Note that the point character '.' between the filename and the filetype is not stored in the directory. Thus the name 'MYPROG.DAT' will be stored as follows:

| Byte No | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Hex Value | 4D | 59 | 50 | 52 | 4F | 47 | 00 | 00 | 44 | 41 | 54 |
| ASCII Value | M | Y | P | R | O | G | | | D | A | T |

**Bytes 13 to 24, file storage pointers**

Data (in the form of files) is stored on disc in 'clusters'. A cluster is simply a block of storage allocated on the disc; it is always an integral number of sectors long. When DOS writers a file to disc it allocates a free cluster and writes the data to this area. However, because of fragmentation of the disc due to file deletion, the original cluster may not be big enough for all of the file to be stored. In this case DOS finds another free cluster and continues storing data there. This process of finding unused 'holes' on the disc and filling them continues until all of the file is stored (or no more holes can be found — DF ERROR!). Thus to recover the data contained in a file we need to know the location and size of each cluster as well the order in which they were allocated. This is what I have called the storage pointers. Each storage pointer contains two entries: 1) a two-byte Logical Sector Number (LSN) of a cluster, and 2) a one-byte cluster size in sectors. The location on the disc of each cluster is given by the LSN. This is a number which starts at zero and increments to the maximum number of sectors on a disc (720 for single sided 40-track discs). It is calculated by multiplying the track number (starting at zero) by the number of sectors per track (18 for DRAGON DOS) and then adding the sector number for that track (starting from zero also). Thus the start of the Dragon DOS directory which is at track 20 sector 3, has an LSN of 362 (20 x 18 + 2). It can be seen that the first sector of the second track has an LSN of 18. As an example let us consider a file which is stored in three clusters as follows:

cluster 1 : 6 sectors starting at track 3 sector 5
cluster 2 : 12 sectors starting at track 39 sector 1
cluster 3 : 4 sectors starting at track 11 sector 18

This gives us the following LSNs:

LSN cluster 1 = 3 x 18 + 4 = 58 (003A HEX)
LSN cluster 2 = 39 x 18 + 0 = 702 (02BE HEX)
LSN cluster 3 = 11 x 18 + 17 = 215 (00D7 HEX)

Since clusters are contiguous areas on the disc the above clusters will cover the following sectors:

cluster 1 : track 3, sectors 5 to 10 inclusive
cluster 2 : track 39, sectors 1 to 12 inclusive
cluster 3 : track 11, sector 18; and track 12 sectors 1 to 3 inclusive

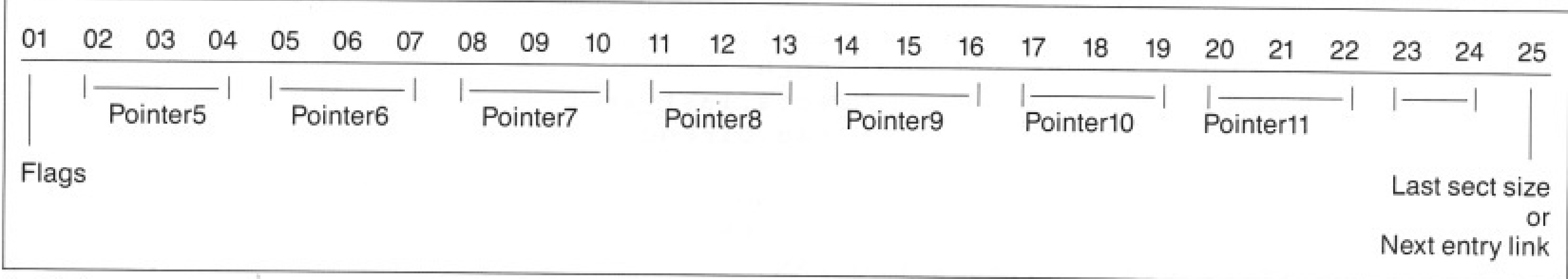
**Byte 25, last sector size**

The above example shows that the file occupies 22 sectors, or 22 x 256 = 5632 bytes on the disc. It does not mean the file is exactly 5632 bytes long! It may be shorter by up to 255 bytes (any shorter than that would require less sectors). In order to keep track of the 'logical' end of the file DOS saves how many bytes are used in the last sector of the last cluster. This value is stored in the last byte of each directory entry. Should the last sector be completely filled then byte 25 will be zero which corresponds to $FF+1 (255+1=256).

Using the above cluster allocation scheme for a file 5600 bytes long called EXAMPLE.BAS, the directory entry would look like:

| 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 18 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 45 | 58 | 41 | 4D | 50 | 4C | 45 | 00 | 42 | 41 | 53 | 00 | 3A | 06 | 02 | BE | 0C | 00 | D7 | 04 | 00 | 00 | 00 | E0 |

Note that Pointer4 (bytes 22-24) contains zeros. This is a flag to show there are no more clusters for this file entry. It can happen however that more than four clusters are required to store the whole file, especially when the disc is heavily fragmented due to frequent use. In this case it is necessary to extend the number of storage pointers for the file slot. This is where the 'Continued' flag of byte 1 comes in. If this bit is set (=1) it indicates that there is another continuation slot for this file somewhere in the directory. That somewhere is given by byte 25 of this slot. Byte 25 then has changed its meaning depending on the state of the continued flag; it no longer contains the number of bytes in the last sector of the file but the slot number of the continuation of this entry. Directory slots are numbered starting from zero, so since each slot occupies 25 bytes then the second slot ( slot number 1 ) starts at the 26th byte of the directory, the third slot at the 51st byte and so on. The second continuation slot may itself be continued which is again indicated by the 'Continued' flag of the first byte of the slot. Since the filename is already stored in the first slot for the file, each continuation slot has a different format to the first: apart from the first and last bytes it contains only storage pointers (6 of them) plus two unused bytes 22 and 23 as shown below:




As before, a pointer containing all zeros signals the end of the list and byte 25 will contain the number of bytes in the last sector, otherwise if the 'Continued' bit is set then byte 25 contains the number of the continuation entry.

When searching through the directory to find files and their cluster pointers we have to be able to distinguish between the first entry for a file containing the filename, and any continuation entry which on its own gives us no information. This is done using the Slot Type flag of the first byte of each slot. If this bit is set then we (and DOS) know it is a continuation slot and we should not attempt to interpret the filename from bytes 2 to 12. It this flag is set then the slot containing it does not contain any useful information but is the logical end of the directory and we go searching no further.

If the Protection flag is set in the first byte of the directory entry this indicates to DOS that the file must not be deleted. This bit is set/cleared by using the PROTECT ON/OFF commands.

One final flag is the 'Valid file' flag which indicates if the file has been deleted or not. A newly created file entry will have this flag cleared (=0). If the file is subsequently deleted using the KILL command then this bit is set to 1, and the bit map of the disc is updated to show that the previously occupied clusters are now free for use. The filename and its associated cluster pointers are not removed from the directory, the flag only indicating to DOS that it may reuse this entry if it wishes. Likewise the contents of the newly released clusters are not touched; they are only flagged as free for reuse. It is this mechanism which allows the 'unKILLing' of files by clearing this flag and re-updating the bit map. It will of course only work if DOS has not written *anything* to disc in the meantime since it is likely to have overwritten all or part of the deleted file.

A newly initialised disc has all its entries set to $89 signifying 1) an invalid file (deleted), 2) the end of the directory, and 3) a continuation slot (contains no useful information on its own).

A word about **file headers**. These are not stored in the directory but in the first nine bytes of the file itself. They have the following format:

| byte | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 |
|---|---|---|---|---|---|---|---|---|---|
| Hex value | 55 | | | | | | | | AA |

LOAD ADDR (bytes 03-04), LEN (bytes 05-06), EXEC ADDR (bytes 07-08), TYPE (byte 02)

Bytes 1 and 9 are used to identify the header; any other values than $55 and $AA will cause DOS to not recognise the header and assume the file is a data file. The presence of a header implies that the file can be loaded directly into memory and provides the data required. Bytes 3 and 4 contain the address at which DOS should start storing the contents of the file into memory. Bytes 5 and 6 give the number of bytes to be loaded ( the length of the file minus the 9 header bytes). Bytes 7 and 8 are the EXEC address of the program (these bytes are put into $9D/$9E to tell BASIC where tostart execution of the program when EXEC is typed). The 'TYPE' byte (byte 2) defines the type of file: 1 means the file is a tokenised BASIC program saved to disc using the SAVE command using only a filename parameter. Should SAVE be used using the extra three parameters to define the addresses as described above then byte 2 of the header will be set to 2 indicating the file to be a machine code file (normally saved by DOS AS 'BIN' files). No other values are defined that I know of.

## Sector bitmap

No discussion of the directory would be complete without a few words about the disc allocation bitmap. This map allows DOS to keep a constant trace of which sectors on the disc are free, and which are used. For each sector on the disc there is a corresponding bit in the bitmap. In the first byte, Bit 0 (the least significant bit) represents LSN 0, and Bit 7 (the most significant bit) represents LSN 7. Bit 0 of the second byte represents LSN 8 and Bit 7 LSN is in use and must not be written to, whereas a bit set to 1 means the sector is free for storing (part of) a file there, DOS automatically clears and sets these bits as it allocates and frees disc sectors when files are created and deleted. It should be noted that because direct sector access using SPREAD and SWRITE does not use the file system, the bitmap is not affected by these commands.

The size of the bitmap varies according to the number of sectors initialised on the disc by DSKINIT. Thus for the default size of a Dragon disc containing 720 sectors we need 720 bits in the map, or 720/8=90 bytes. This fits easily into one sector. However if the disc is initialised for 80 tracks and 2 sides (36 sectors per track) the number of sectors is 80x36=2880. This will require 360 bytes in the bitmap — more than one sector! Dragon DOS reserves enough space (2 sectors) for these large bitmaps in the first two sectors of the directory track, ie track 20, sectors 1 and 2 (LSN 360 & 361). It should be noted that the last four bytes of the first sector of the bitmap have a special meaning:

| Byte | Default value (hex) | Meaning |
|---|---|---|
| 253 | 28 | Number of tracks on the disc |
| 254 | 12 | Number of sectors per track |
| 255 | D7 | Complement of byte 253 |
| 256 | ED | Complement of byte 254 |

**The Basic09 program ReadDD**

This program uses all the information outlined above except the bitmap. It was written to enable the transfer of any file from Dragon DOS to 0S-9. Should the Dragon DOS file be a Basic program or a memory image file (binary file) then the file load information is displayed and the user given the option of removing the file header from the 0S-9 file. If this is done then a new file with 'noheader' added to the name is created. Thus the user has the option of keeping both versions on his/her 0S-9 disc. A data file which has no header information is copied 'as is'. Should the user possess only one drive then the option is given to use this drive to do both reading from a Dragon disc and writing to an 0S-9 disc. This will mean a lot of disc exchanging in response to the prompts but at least a transfer is possible.

Owing to an 'error' in the distributed device descriptors for /DO and /D1 in 0S-9, DRAGON discs cannot be read directly without a little playing with the device descriptor. This is because the descriptors supplied by Microware define the discs as having evrery file allocated space in chunks (clusters) of 8 sectors at a time whereas Dragon discs are allocated 1—sector chunks. This can be overcome by 'patching' the descriptor for the drive in which you are to read the Dragon disc in the following way:

```
0S-9:debug      * Call the OS-9 debugger
DB: L D1        * Locate the device descriptor
DB: . .+20      * Open offset $20 (Note the space between the two points)
DB: =1          * Cluster size = 1
DB: q           * Exit the debugger
```

The above five lines (without the prompts) can be edited into a file and called as a command file from the keyboard or the 0S-9 startup file in which case the patch will be done automatically whenever 0S-9 is booted. It is also possible to change the descriptor permanently whether it is in the boot file or a separate descriptor file, but this is best left to the 0S-9 'hackers'. (If you want to contact me and don't mind international telephone calls which are cheaper after 20:30 UK times give me a ring on 010-33-76352828 and I will gladly tell you how it's done). If the drive used for reading the Dragon disc is /DO then the first command to the debugger would be 'L DO' instead of 'L D1'. In any case the device descriptor must be in memory at the time of the patch (use LOAD if not), and it does not upset 0S-9 to patch every descriptor in this way since 0S-9 normally gets its allocation information from the disc itself if there is a valid 0S-9 disc in the drive. This patch is only necessary when doing a 'raw' disc read using the '@' character after the device name as is done in ReadDD. One other step is required to ensure fault-free reading of a non-0S-9 disc; a valid 0S-9 disc must be in the drive for the very first access of the named device. Thus after patching the device descriptor as described above you should place an 0S-9 disc in the drive and then access it in some way, such as:

```
0S9:chd /d1
or
0S9:dir /d1
```

The 0S-9 disc can then be removed from the drive and replaced with the Dragon disc. There should be no problem in reading it from then on.

```
PROCEDURE readdd
 0000      (*
 0003      (* Programme to transfer a DRAGON-DOS file to OS9.
 0035      (*
 0038      (* Written by P.A.Dagleish 7-Jan-87
 005B      (*
 005E      REM Define our special types for easy directory access
 0093      TYPE Pntrtype=LSN:INTEGER; Nsectors:BYTE
 00A4      TYPE DirEntry=flags:BYTE; Name:STRING[8]; Ext:STRING[3]; Pntrs
             (4):Pntrtype; Last:BYTE
 00D8      TYPE Exttype=eflag:BYTE; Epntr(7):Pntrtype; unused:INTEGER; Elast
             :BYTE
 00FC      TYPE Header_Type=First_byte,ftype:BYTE; Load_Addr,Length,EXEC_Addr
             :INTEGER; Last_Byte:BYTE
 011F      (* Allocate variables using these special types *)
 0151      DIM File,Directory(160):DirEntry
 0163      DIM Extension:Exttype
 016C      DIM pntr:Pntrtype
 0175      DIM header:Header_Type
 017E      (* Now define the simple variables *)
 01A3      DIM First_read,Got_Header:BOOLEAN
 01AE      DIM OS9Name:STRING[12]
 01BA      DIM Ddrive,OS9drive:STRING[1]
 01CA      DIM OS9chan,DDn,i,j,k:INTEGER
 01E1      DIM Filenum,Track,Sect,NSects,last_sect_size:INTEGER
 01F8      DIM BASIC,BINARY,Head,Locked,DirEnd,Continued,Valid:BYTE
 0217      DIM Pad:STRING[8]
 0223      DIM Dummy,Reply,FF,NULL:STRING[1]
 023B      DIM NumonLine,MaxonLine:INTEGER
 0246      DIM Single,Done,More:BOOLEAN
 0255      DIM Buffer(256),Onebyte:BYTE
 0265      DIM Flen,Fsize,Position,Bufsize:REAL
 0278      (* Initialise some useful constants *)
 029E      Bufsize:=SIZE(Buffer)
 02A9      BASIC:=1
 02B0      BINARY:=2
 02B7      MaxonLine:=3
 02BE      NumonLine:=0
```

```
02C5       NULL:=CHR$(0)
02CD       FF:=CHR$($FF)
02D6       Pad:="          "
02E5       Head:=$01
02ED       Locked:=$02
02F5       DirEnd:=$08
02FD       Continued:=$20
0305       Valid:=$80
030D       Flen:=.0
0318       (* ------------------ Start of Main Programme -------------- *
0356       INPUT "Drive number of DRAGON disc? (Default=1):",Ddrive
0387       IF Ddrive="" THEN
0393         Ddrive:="1"
039B       ENDIF
039D       OPEN #DDn,"/D"+Ddrive+"@":READ
03B2       SEEK #DDn,0
03BB       INPUT "Drive number of OS9 disc? (Default=0):",OS9drive
03E9       IF OS9drive="" THEN
03F5         OS9drive="0"
03FD       ENDIF
03FF       CHD "/d"+OS9drive
0409       Single:=Ddrive=OS9drive \(* Single drive transfer ? *)

0432       Track:=20
0439       j:=1
0440       IF Single THEN
0449         PRINT "Insert DRAGON source disc, then hit a key";
0477         (* Wait till he types something on the standard input path *)
04B4         GET #0,Dummy
04BD         PRINT
04BF       ENDIF
04C1       (* FOR loop reads the whole of the DRAGON directory into our array st
ructure "Directory" *)
051C       FOR Sect:=2 TO 17
052C         Position:=(Track*18+Sect)*256.
0543         SEEK #DDn,Position
054D         FOR i:=1 TO 10
055D           GET #DDn,Directory(j)
056B           j:=j+1
0576         NEXT i
0581       NEXT Sect
058C       j:=1
0593       (* The WHILE loop prints out the valid filenames in the directory tog
ether with their entry number *)
05F8       WHILE LAND(Directory(j).flags,DirEnd)=0 DO
060E         FOR i:=1 TO 8
061E           (* Basic09 requires strings to be terminated by $FF *)
0654         EXITIF MID$(Directory(j).Name,i,1)=NULL THEN
066D           Directory(j).Name:=LEFT$(Directory(j).Name,i-1)+FF
068D         ENDEXIT
0691         NEXT i
069C         File:=Directory(j)
06A7         (* Only print out slots which are valid header entries *)
06E0         IF LAND(File.flags,Valid+Head)=0 THEN
06F7           PRINT USING "i3>,'-',s8<,'.',s3>",j,File.Name,File.Ext,

0722           NumonLine:=NumonLine+1
072D           IF NumonLine>=MaxonLine THEN
073A             PRINT
073C             NumonLine:=0
0743           ENDIF
0745         ENDIF
0747         j:=j+1
0752       ENDWHILE
0786       PRINT
```

```
0758        (* Ask which file is required *)
0778        REPEAT
077A          REPEAT
077C            INPUT "Select number of file to copy: ",Filenum
07A3          UNTIL Filenum<j AND Filenum>=0
07B6          IF Filenum=0 THEN
07C2            END "Abort"
07CB          ENDIF
07CD        UNTIL LAND(Directory(Filenum).flags,Valid+Head)=0
07E6        File:=Directory(Filenum)
07F1        OS9Name:=File.Name+"."+File.Ext
0807        IF Single THEN
0810          PRINT  \ PRINT "Insert OS9 destination disc, then hit a key"
             ;
0842          GET #0,Dummy
084B        ENDIF
084D        CREATE #OS9chan,OS9Name:UPDATE
0859        More:=LAND(File.flags,Continued)=1
086B        First_read:=TRUE
0871        Got_Header:=FALSE
0877        (* Handle the 1st entry slot *)
0896        FOR i:=1 TO 4
08A6          pntr:=File.Pntrs(i)
08B4          IF i<4 THEN
08C0            Done:=File.Pntrs(i+1).Nsectors=0
08D7          ELSE
08DB            Done:=NOT(More)
08E4          ENDIF
08E6          NSects:=pntr.Nsectors
08F1          IF Done THEN
08FA            NSects:=NSects-1
0905          ENDIF
0907          Position:=pntr.LSN*256. \(* NOTE multiply by a real *)
0937          SEEK #DDn,Position
0941          (* Read any possible header then reread the whole of the first sect
or *)
0989          IF First_read THEN
0992            First_read:=NOT(First_read)
099B            GET #DDn,header
09A5            (* Re-position to start of the file (header) *)
09D4            SEEK #DDn,Position
09DE            Got_Header:=header.First_byte=$55 AND header.Last_Byte=
             $AA
09F8          ENDIF
09FA          RUN readsect(DDn,OS9chan,NSects,Flen,Buffer,Single)
0A1D        EXITIF Done THEN  \ ENDEXIT
0A2A        NEXT i
0A35        (* Now handle the extension entries if any *)
0A62        IF NOT(Done) THEN
0A6C          Done:=LAND(Directory(Filenum).flags,Continued)=0
0A81        ENDIF
0A83        IF Done THEN
0A8C          last_sect_size:=File.Last
0A97        ELSE
0A9B          Filenum:=Directory(Filenum).Last
0AA9          REPEAT
0AAB            Extension:=Directory(Filenum)
0AB6            More:=LAND(Extension.eflag,Continued)=1
0AC8            FOR i:=1 TO 7
0AD8              pntr:=Extension.Epntr(i)
0AE6              (* Check if this is the last pointer in the list *)
0B19              (* i.e. the next one is zero or if this is the 7th *)
0B4E              (* is there a continuation entry? *)
0B72              IF i<7 THEN
0B7E                Done:=Extension.Epntr(i+1).Nsectors=0
0B95              ELSE
```

```
0B99          Done:=NOT(More)
0BA2        ENDIF
0BA4        NSects:=pntr.Nsectors
0BAF        IF Done THEN
0BB8          (* If we're at the end only read NSects-1 complete sectors *)
0BF5          (* since the last one is normally only partly filled *)
0C2C          NSects:=NSects-1
0C37        ENDIF
0C39        Position:=pntr.LSN*256.
0C4C        SEEK #DDn,Position
0C56        (* Read NSects sectors *)
0C6F        RUN readsect(DDn,OS9chan,NSects,Flen,Buffer,Single)
0C92            EXITIF Done THEN  \ ENDEXIT
0C9F            NEXT i
0CAA            IF Done THEN
0CB3              last_sect_size:=Extension.Elast
0CBE              (* If finished then the last byte gives us the number *)
0CF6              (* of bytes to read in the last sector, otherwise the *)
0D2E              (* last byte points to the continuation slot *)
0D5D              (* and the REPEAT loop continues *)
0D80            ELSE
0D84              Filenum:=Extension.Elast
0D8F            ENDIF
0D91          UNTIL Done
0D99        ENDIF
0D9B        (* Here we read the last (partial) sector *)
0DC7        RUN readpartsect(DDn,OS9chan,last_sect_size,Flen,Buffer,Single
             )
0DEA        CLOSE #DDn \(* Finished with DRAGON Disc *)
0E0F        (* Here we print the info found in any file header *))
0E45        IF Got_Header THEN
0E4E          IF header.ftype=BASIC THEN
0E5E            PRINT "File is a BASIC program file"
0E7E          ELSE
0E82            IF header.ftype=BINARY THEN
0E92              PRINT "File is a binary image file"
0EB1            ENDIF
0EB3          ENDIF
0EB5          PRINT USING "'Load address: $',h4,'",header.Load_Addr;
0ED8          RUN praddr(header.Load_Addr)
0EE5          PRINT USING "'Load length : $',h4",header.Length;
0F06          RUN praddr(header.Length)
0F13          PRINT USING "'EXEC address: $',h4",header.EXEC_Addr;
0F34          RUN praddr(header.EXEC_Addr)
0F41          (* Allow the removal of the header if required *)
0F72          INPUT "Remove the 9-byte header from the OS9 copy? [Y/N]"
               ,Reply
0FAB          IF Reply="Y" OR Reply="y" THEN
0FC0            CREATE #DDn,OS9Name+"_noheader":WRITE
0FD8            Fsize:=Flen-9.
0FE7            SEEK #OS9chan,.9
0FF4            WHILE Fsize>=256. DO
1004              GET #OS9chan,Buffer
100E              PUT #DDn,Buffer
1018              Fsize:=Fsize-Bufsize
1024            ENDWHILE
1028            WHILE Fsize>.0 DO
1038              GET #OS9chan,Onebyte
1042              PUT #DDn,Onebyte
104C              Fsize:=Fsize-1.
105B            ENDWHILE
105F            CLOSE #DDn
```

```
 1065        ENDIF  \(* End of header removal *)
 1083       ENDIF
 1085       CLOSE #OS9chan
 1088       PRINT
 108D       END "Transfer complete"
PROCEDURE readsect
 0000       (* Read N sectors from "in", starting at present position *)
 003C       (* Output goes to "out" in "buffer"_size chunks *)
 006E       PARAM in,out:INTEGER; N:INTEGER; flen:REAL; buffer(256):BYTE
 0090       PARAM single:BOOLEAN

 0097       DIM i:INTEGER
 009E       DIM dummy:STRING[1]
 00AA       i:=N
 00B2       WHILE i>0 DO
 00BE         IF single THEN
 00C7           PRINT  \ PRINT "Insert DRAGON source disc, then hit a key"
                 ;
 00F7           GET #0,dummy
 0100         ENDIF
 0102         GET #in,buffer
 010C         IF single THEN
 0115           PRINT  \ PRINT "Insert OS9 destination disc, then hit a key"
                 ;
 0147           GET #0,dummy
 0150         ENDIF
 0152         PUT #out,buffer
 015C         i:=i-1
 0167       ENDWHILE
 016B       flen:=flen+256.*N
PROCEDURE readpartsect
 0000       (* Read N bytes from "in", starting at present position *)
 003A       (* Output goes to "out" in N_size chunks *)
 0065       PARAM in,out,N:INTEGER; flen:REAL; buffer(256):BYTE
 0085       PARAM single:BOOLEAN
 008C       DIM bite:BYTE; Nbytes,i:INTEGER
 009D       DIM dummy:STRING[1]
 00A9       IF single THEN
 00B2         PRINT  \ PRINT "Insert DRAGON source disc, then hit a key"
               ;
 00E2         GET #0,dummy
 00EB       ENDIF
 00ED       GET #in,buffer
 00F7       Nbytes:=N
 00FF       IF N=0 THEN
 010B         Nbytes=256
 0113       ENDIF
 0115       IF single THEN
 011E         PRINT  \ PRINT "Insert OS9 destination disc, then hit a key"
               ;
 0150         GET #0,dummy
 0159       ENDIF
 015B       FOR i:=1 TO Nbytes
 016C         bite:=buffer(i)
 0177         PUT #out,bite
 0181       NEXT i
 018C       flen:=flen+Nbytes
PROCEDURE Praddr
 0000       PARAM address:INTEGER
 0007       temp:=address
 0010       WHILE temp<0 DO
 001D         temp:=temp+65536.
 002C       ENDWHILE
 0030       PRINT " ("; temp; ")"
```

New software for review should be sent to *Dragon User*, 12-13 Little Newport Street, London WC2H 7PP.

# Fast characters

**Program:** *Hi-Text*
**Supplier:** Microvision Software, 8 Wensley Road, Liverpool L9 8DW.
**Price:** £5.00 (cassette)

I have recently purchased a copy of *Hi-text* from Microvision. I was amazed by the speed at which this program prints characters on a TV screen.

*Hi-text* offers 64 x 21 display. Unlike many others *Hi-text* displays which define characters on a 4 x 8 matrix (giving illegible 64 column text), *Hi-text* defines characters on a 3 x 8 matrix (with the last row of the matrix left for decenders), giving crystal clear display even on an ordinary black and white TV with contrast and brightness properly tuned up. In fact, a 1mm gap is measurable between two 'H's on a 14" TV.

*Hi-text* offers a screen printing speed matchable to that of the Amstrad CPC *Masterfile* when used from within Dragon Basic.

Other features of Hi-text include user definable character facilities. A formula has been included for this purpose. There are four control codes which may be sent to the driver using the PRINT CHR$(Z) command. The effects generated are as follows:

| Value of Z | Effects |
|---|---|
| 12 | Clear Screen |
| 8 | Backspace |
| 7 | Beep |
| 1 | Inverse/True Video toggle |

Being in 64 columns, the PRINT@ command has to be replaced by PRINT X,Y, where X has a value ranging from 0-63 and Y, 0-191. The CLS command when used with any number will clear the screen (except with values greater than 8 which produce the normal copyright message). Full Auto-repeat of keys has been implemented. You may have black text against white or green background. Machine code users may use any of the Basic ROM routines to output text such as $B54A or $90E5. The program takes up only about 2.5K when installed, which is a relatively low figure.

The only drawback of *Hi-text* is that you cannot use it in the 64K mode if you have a Dragon 64.

Many serious business software packages such as word-processors, report generators and bank accounts can be written with the aid of this program. The first advantage is that you now have a screen which displays information two and a half times more than before. Secondly, 64 characters per line of text comfortably matches the width of an A4 size paper. This means that many programs can now be designed on a What You See Is What You Get basis. In fact, the popular *Tasword* and *The Writer* for the Spectrum are two such programs which take advantage of 64 characters.

Those of you have a Dragon 64 may incorporate those Auto-repeat POKE statements (given in Dragon's Supplementary Manual) into *Hi-text* (to do this you must first install *Hi-text*. The result is that you have a fast response keyboard in addition to a fast screen. Words will not be left out when you type fast. The 'smoothness' of the Auto-repeat action is however affected, because of the already built in Auto—repeat routine in *Hi-text*. Microvision should consider adding a fast keyboard routine to *Hi-text*, such as those found in Microdeals *Telemod*.

At a cost of £5.00, *Hi-text* gives your Dragon a fast, business-usable kind of screen; this represents good value for money.

Finally, I would appreciate if Microvision would consider producing a 64 characters display wordprocessor which offers facilities and standards similar to those found in *Tasword* or *The Writer*, such as complete on-screen formatting (this includes justification of text, word wrapping, automatic page breaks, centring line and sideways scrolling 128 column text) and others like header, footer and a second character set for the printer.

*Dr. Siew Yin Hong*



# Write: ADVENTURE

***Peter Gerrard** begins a news series on creating your own games*

OUTSIDE my window it is another dismal day and the streets of North London are paved with objects of a dubious nature and certainly not gold. But do I care? Not a jot. For I have just returned from a voyage of discovery on the inside of a strange satellite deep in space, trying to deliver a mysterious box to a mad professor (*So that's what became of the Boulder Crashes — Ed.*) and dodging all kinds of unusual and unnatural obstructions, aided only by my trusty IMP. Not your ordinary companion, is Eddie the IMP. He is a small transmitter/receiver, fitting comfortably under the skin but apt to panic in moments of crises.

All this is part of an adventure called *Take Two* for some nefarious reason that I will not go into here. Written it was by yours truly, and over the course of the next few months I'm going to be sharing some of my adventure writing secrets with you through the pages of this august (*May this month — Ed.*) journal. The object of all this is of course to get you writing your own adventures. I know from my brief acquaintance with the Adventure Trail column that a number of Dragon readers already write their own games, but a much larger number do not and wish to know where to start. Read on.

## Why Bother?

Utilities exist to aid you in the writing of adventures, so why not use those? Well, for one thing you will learn absolutely nothing about programming, and for another you will only be producing something that relies almost totally on the skill of someone else. Yours will be the imagination, but the author of the utility will be the one that deserves the most credit. Surely it would be better if you'd written the whole thing yourself? Then you could sell it as your own creation from start to finish, without bothering about such murky things as copyright. You would also have, let's be honest, the immense sense of pride that comes from completing your own game, written in your own way.

I will be providing some programming utilities, you will be doing the actual work, altering those utilities, changing them to fit into your own games and your own particular scenarios. Or, to put it another way, I will provide the covers and numbered pages of a book, you will fill in the pages with whatever you want.

I can still remember the amazing sense of pleasure that I got on completing my first Dragon adventure and receiving a query from someone who couldn't solve a particular problem. It was only a small game, with a limited number of puzzles and short term descriptions, but I still get reminded of it from time to time. Some people say it was the best thing I've ever written. You definitely can't win'em all. But between you, me and the Dragon, I think we're capable of making more than a few people sit up and take notice. I see from some of the letters received in *Adventure Trail* that quite a few husband and wife, boyfriend and girlfriend teams of adventure players exist. How about writing an adventure for your companion? Greater love hath no person, etc. etc.

## Getting started

In order to think about writing an adventure game, you need to have no programming skills whatever. This is not meant to deride people who produce the ideas rather than the programs, but it is a statement of fact. The greatest programmer in the world will not produce a decent adventure without a very strong storyline behind it, and if someone else provides the storyline, all well and good.

And if we want to start thinking about a possible scenario for our game, let us first of all begin by looking at what we won't be doing.

There will be no (or very few) graphics in the adventure-writing technique I am going to concentrate on. For one thing I do not like graphic adventures, and although I can admire other people's computers running glorious pictures from games like *The Pawn*, it doesn't make me want to rush to the keyboard and produce something similar. Even if I could, which I doubt. I much prefer text only games, on the premise that the most vivid imagination, the most superb illustrations, the best of artistic designers, will not compete with the images conjured up inside my own head by a good, detailed, atmospheric description of the game world.

We will not be able to have adventures with three million locations, and a practical limit will be around one hundred. That will suffice to concoct a realistic arena in which to play the game, and too much wandering about without doing anything seems to defeat the point of an adventure, to my way of thinking. There should be problems to solve, not vast underground corridors to explore that go on and on seemingly without end.

We will not have a parser (the part of the program that provides the vital link betwen player and computer, and which interprets what the player has typed in) capable of understanding the likes of "turn to page 124 of the red book and read the third paragraph before putting it back under the table". Forget it. If you had spelt the first word incorrectly it would all be a waste of time anyway, so the most advanced thing that we'll be looking at will be something like "look under the clear window", "climb over the third door", "open the canister with the gloves", etc. Enough, in other words, to produce some fairly detailed problems without resorting to the obscurely bizarre.

Right, that's what we can't do. Beyond that the world is your barnacle for the time being, and we're aiming for a lot of text, a reasonably sophisticated parser, and a rather large vocabulary. Time to sit down in front of some paper.

## Mapping out

Before touching the computer and before next month's issue of *Dragon User* appears and we start doing some programming you'll need to have a few things sorted out. Where is the adventure going to take place? Outer space? Middle Earth? The American Wild West? Surbiton? Up to you, but do get some idea of where the action will be happening and the type of game world in which your player will be operating.

Having decided where we're going to be, what will be the object of the adventure? Survival? Collecting treasures? Completing a dangerous mission? Getting out of bed and having breakfast? Again, it's up to you, but try and sort that out as well. Only roughly, it doesn't matter yet.

Next, draw a map on a sheet of paper. Start with, say, fifty locations, because you can always add more later. Just draw the numbers one to fifty on a sheet of paper, and that will become your adventure world. As a brief example, like this:

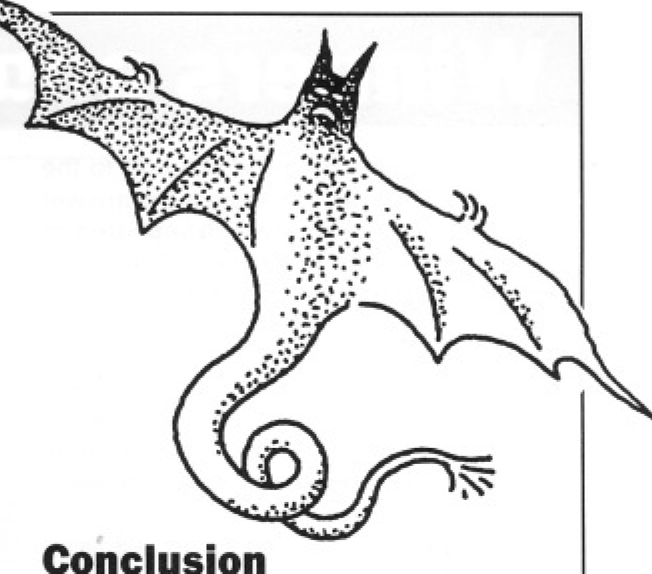
```
1-2-3-4-5
  |
  6-7-8-9-10
  |
  11
```

and so on. Assuming north to be at the top of the page, room one heads east to room two, which heads east to three and south to six. Room six heads north to two, south to eleven, and east to seven. Any number that links up with another one means that the two rooms will be connected. Now you'll want to set up some problems, doesn't matter

where they are yet, as long as we know where we're going. Try and stick to an average of one problem to every three or four locations, giving the player the chance to explore a bit before getting stuck. In the example above you might circle the numbers 2 (can't go anywhere before solving that), 4, 6, and 8: all useful looking locations that bar progress to other ones.

Say we're going to be inside a satellite in outer space, and locations 2, 3, 4, 6, 7 and 8 obviously appear to form a complete area. They could be, for instance, the control area in which the player starts his mission. Location 10 looks like a corridor going off somewhere, while location 11 could be heading off into the heart of the satellite. Start labelling areas on your complete map, and pretty soon you'll have about nine or ten distinct areas for your fifty location adventure (it will grow, believe me). Knowing what's going to be there helps you to start thinking about problems. Location 2 could be the entrance to the control room, a locked door which you would obviously need to get past in order to start the game. A problem straight away, how to open the door. It doesn't take long for the whole game to build up from your simple scrawl on paper.

It is also worth drawing a tidier map, as I know to my cost. I once designed a massive adventure game using discs that occupied hundreds of locations, and had well over sixty involved problems to solve. Looked great on my vast, badly drawn map. It was also unsolvable, because I'd got two locations numbered the wrong way around, and the player was doomed to go round in circles for evermore. Putting that problem to rights took up an awful lot of time that I needn't have had to waste if I'd got a tidier, legible map to look at.



## Conclusion

The Dragon is lurking away in the corner, waiting to get started, and next month we will be powering up and starting to produce the first part of our adventure game. Until them, get that map drawn, think about all the problems that the player might have to come across in your imaginary world, and label them. Clearly! Make sure that the game arena forms a coherent whole, and that there are no obvious flaws in it, no problems that are just so ludicrous that nobody will ever solve them. Get the game areas sorted out, so that you know roughly what every location corresponds to.

Finally, start thinking about the sort of words that the player might have to type in in order to go through the completed game. A vocabulary list in other words, because that is what we will be looking at next time. Bye for now.

# National Dragon Users' Group

*Introduction by* ***Philip Beed***

HAVING written a favourable text on *Dragon User* in *Dragon Update* it's now my turn to do the same thing for the NDUG in DU. Many people will have noticed that advertising for the group is very low key. This is deliberate policy, as the number of user groups which took money and then vanished without delivering any goods has meant that people are now very wary of such groups. So the National Dragon User Group is run more on a personal recommendation basis, and our high membership which is still growing must speak well for the group.

NDUG is the largest Dragon user group in existence, with members in the UK and abroad. It has been running for well over two years now and is likely to be with us for as long as there are Dragon owners who feel the need for it. While it is not a necessity for a Dragon owner to join, I would urge them to give it every consideration. A strong core of users can only help us to get more out of our micros.

Membership is currently £7.50 for a year, virtually unchanged since day one. For this you get a monthly newsletter, *Update*, which tries to give information from all spheres of interest in the group, so there is more technical and news and views content than games information. More important than the newsletter, though, are the group's other facilities. Tapping into the expertise of members, the group can aim to answer questions on practically any subject; help can be only a letter or a phone call away. If no-one has actually solved your problem already, then some-one will be expert enough to investigate it. How to buy a disc drive? Screen dumps for a specific printer? Programming difficulties? How to upgrade to 64K? etc.

There is also a full page of classified ads. which are free to users, through which some good buys can be had. The group has issued some programs of its own at a very reasonable cost: a Forth operating system, a disc editor, and a graphics utility, and circuit diagrams for 32s and 64s, as well as the DOS, are available, as well as a detailed manual on how to do your own 32 to 64 upgrade. An at-cost repair service is also available for computers and peripherals, an excellent way of beating the cowboy operators often heard about (*You chaps should have mentioned that before — Ed.*). Just recently the group has acquired pages on Micronet (page 700110037), a graphics library has been started, and a direct helpline for various problems has been started.

The most important point about NDUG is that it is a group not a publication, and anyone joining to receive a monthly newsletter may well be disappointed. You will get out what you put it. Everyone no matter what level they are at, or what their interests are, has something to contribute to the group and if they do this they will find that by taking part and helping others they will benefit themselves. I have been a member for 18 months now and wish I had joined earlier. The group has helped me to spend my money wisely on improving my system, has given me an outlet for my feelings on various topics, and helped me to expand my knowledge of the Dragon. I would highly recommend it to anyone who owns a Dragon.

# Winners and Losers

**Every month, Gordon Lee will look at some prize programming points from a previous month's competition**

BEFORE considering the solution to the Christmas competition, here is the answer to the tile-placing game which appeared on December's competition page. The game, for which a listing was given, required sixteen coloured tiles to be placed on a four by four grid such that a tile was not placed on, or adjacent to, a square of its own colour. If the squares of the grid are numbered in sequence from 1 (top left) to 16 (bottom right), one possible solution is as follows:

Place orange on 16, cyan on 12, yellow on 15, green on 11, red on 14, green on 2, red on 6, green on 4, red on 8, orange on 10, cyan on 13, orange on 1, cyan on 5, orange on 3, and yellow on 9.

Now to the competition, which was in the form of an alphamatic in which the digits, 0 to 9, had to be assigned to the letters A, F, H, M, N, O, R, S, T and W, such that SANTA is prime, FROST is a multiple of 7, NORTH is a multiple of 11, and SNOWMAN is a perfect square.

My own solution, as given on page 27 of the March edition, utilises three separate listings — one each to determine possible values for 'SNOWMAN', 'SANTA', and 'FROST'. I mentioned in the March *Winners and Losers* column that reader Alan Thomas suggested that techniques which are not a hundred per cent programming are a bit of a cheat, so I must imagine that he must regard my solution, with its manual cross-checking, as an out-and-out swindle! However, I still maintain that it is perfectly valid to extract information 'by hand' from a number of separate (and simpler) programs. For example, having tested all possible values for 'SNOWMAN', we know that N must equal 6, and S can only be either 1, 2, 4, or 9. If these facts are then programmed in to subsequent listings, quite a reduction in running time can be effected. There is also much less chance of an error creeping in to a simple program than with a more complex one. This is a point that has been raised from a number of letters from readers who are still learning the basics of programming, and who ask for the problems (and their solutions) to be made more simple.

As a general principle, perhaps the best rule is to aim for a method of solution that provides the answer in the *shortest* time, ie programming time *plus* running time. However, for readers who are obsessive about these things, I have included this program from Terry Potter which solves the competition problem in one fell swoop. Here, each section of the program tests for one of the word substitutions in turn, and only values which pass each particular test ae sent on (using a GOSUB instruction) to the next section. Of all the entries submitted, this program was probably the clearest in terms of layout and even comparative novices to programming should be able to follow the method used. Note the subroutine at line 4000 which creates an array containing the primes from 2 to 313 which are later used at lines 3010-3030 to test for the primality of 'SANTA'. My only comments on this listing would be to include the extra statement STEP 2 at the end of line 4020, since we are only interested in odd numbers as possible primes. Also, I would remove the STOP command at line 3050 to allow the program to run on to test for the possibility of any alternative solutions.

A number of entrants used slightly differing techniques. Phil Sapiro used a program to find possible values for 'FROST', 'NORTH', and 'SNOWMAN'. This, he reports, produces just one possible solution, from which the value of the prime 'SANTA' is forced. Consequently, it is not necessary to include a prime check in the listing at all.

G. R. Barber of Sutton Coldfield used a similar method, though including a prime check routine as a final confirmation. Mr. Barber also asks for clarification of the solution to the September competition, as given on page 30 of the December *Dragon User*. This was due to a typing error, as you are probably now aware from February's *Winners and Losers*, where the correct solution is given. Sorry, folks, for any confusion caused!

Finally, my own personal favourite tiebreaker was from Keith David, who wrote: "I want to be perfectly frank, my mind has gone totally blank, the tiebreaker's too tough, though I've tried hard enough, I think Gordon Lee is a crank".

Now, how did he know that???

```
20 DIMC(90)
30 GOSUB 4000
110 FOR Y=1012 TO 3142
120 W$=STR$(Y*Y)
130 IF VAL(MID$(W$,3,1))<>VAL(MID$(W$,8,1)) THEN 270
140 IF VAL(MID$(W$,3,1))=0 THEN 270
150 S=VAL(MID$(W$,2,1))
160 N=VAL(MID$(W$,3,1))
170 O=VAL(MID$(W$,4,1))
180 W=VAL(MID$(W$,5,1))
190 M=VAL(MID$(W$,6,1))
200 A=VAL(MID$(W$,7,1))
210 IF A=0 OR A=2 OR A=4 OR A=5 OR A=6 OR A=8 THEN 270
220 IF S=N OR S=O OR S=W OR S=M OR S=A THEN 270
230 IF N=O OR N=W OR N=M OR N=A THEN 270
240 IF O=W OR O=M OR O=A THEN 270
250 IF W=M OR W=A THEN 270
260 IF M=A THEN 270
265 GOSUB 1000
270 NEXT
1000 V=(N*10000+O*1000)/11
1010 K=INT(V)*11+11
1020 FOR X=K TO (K+990) STEP 11
1030 X$=STR$(X)
1040 R=VAL(MID$(X$,4,1))
1050 T=VAL(MID$(X$,5,1))
1060 H=VAL(MID$(X$,6,1))
1070 IF R=S OR R=N OR R=O OR R=W OR R=M OR R=A OR R=T OR R=H THEN 1110
1080 IF T=S OR T=N OR T=O OR T=W OR T=M OR T=A OR T=H THEN 1110
1090 IF H=S OR H=N OR H=O OR H=W OR H=M OR H=A THEN 1110
1100 GOSUB 2000
1110 NEXT X
1120 RETURN
2000 F=1
2010 IF F=S OR F=N OR F=O OR F=W OR F=M OR F=A OR F=K OR F=T OR F=H
THEN F=F+1:G OTO2010
2020 P=(F*10000+R*1000+O*100+S*10+T)/7
2030 IFP=INT(P) THEN GOSUB 3000
2040 RETURN
3000 SA=S*10000+A*1001+N*100+T*10
3010 FOR Q=1 TO B
3020 IF SA/C(Q)=INT(SA/C(Q)) THEN 3060
3030 NEXT Q
3031 PRINT" S  N  O  W  M  A  N"
3032 PRINTS;N;O;W;M;A;N
3033 PRINT
3034 PRINT" N  O  R  T  H"
3035 PRINTN;O;R;T;H
3036 PRINT
3037 PRINT" F  R  O  S  T"
3038 PRINTF;R;O;S;T
3039 PRINT
3040 PRINT" S  A  N  T  A  "
3041 PRINTS;A;N;T;A
3050 STOP
3060 RETURN
4000 B=1
4010 C(1)=2
4020 FOR Z=3 TO 314
4030 FOR D=1 TO B
4040 IF Z/C(D)=INT(Z/C(D)) THEN4090
4050 NEXT D
4060 B=B+1
4070 C(B)=Z
4090 NEXTZ
4100 RETURN
The solution is:-SNOWMAN=1653796. NORTH=65802. FROST=48510. SANTA=19609.
```

# Ports and pins

I HAVE just bought a Dragon 64 without a user manual, and I do not know how to use the serial port. What are the pins and how do I use this port from programs?

*Paul Smith*
*Birchwood Road*
*Penylan*
*Cardiff*

**THERE seems to be quite a healthy trade in second hand Dragon 64s at the moment. A number of readers have asked where they can get the 'Dragon 64 Supplement', as this contains information on using the serial port. Unfortunately, as far as I know it is no longer available from any official source, so I will try to answer all these queries in one go. Here is all the useful information from the elusive booklet:**

**The Baud rate of the serial port is set from basic using the following:**

```
POKE &HFF07,(PEEK(&HFF07) AND
240) OR X
```

**where 'X' holds the code for the required baud rate, one of the following:**

**1=50, 2=75, 3=110, 4=135, 5=150, 6=300, 7=600, 8=1200, 9=1800, 10=2400, 11=3600, 12=4800, 13=7200, 14=9600.**

**The format of serial data is 1 start, 8 data and 2 stop bits with no parity.**

**To select between printer ports use:**

**Serial port — POKE 1023,1**

**Parallel port — POKE 1023,0**

**Locations 1021/1022 hold a 16bit end of line delay (in 10 millisecond units) for the serial printer port.**

**To send a character to the serial port directly, use:**

```
10 IF (PEEK(&HFF05) AND 16)=0
THEN 10
20 POKE &HFF04, ASC(C$): REM C$
IS CHARACTER TO SEND
```

**To read in a character from the serial port use:**

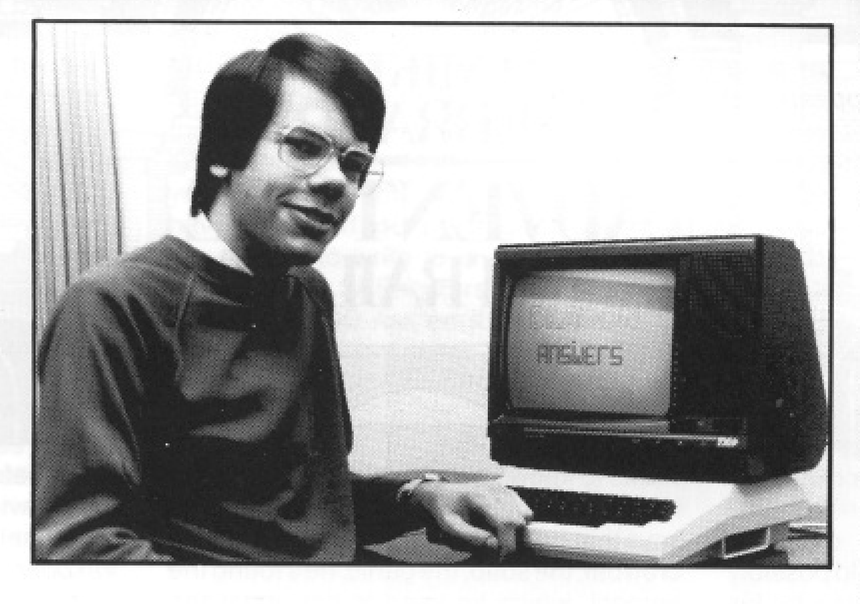
```
10 IF (PEEK(&HFF05) AND 8)=0
THEN 10
20 C$=CHR$(PEEK(&HFF04))
```

**The pin assignment of the port is as follows:**

**pin 6=TX Data, pin 1=GND, pin 4=DTR, pin 2=RX Data, pin 5=CTS, pin 3=+12V, pin 7=-12V.**

**Finally, to use the autorepeat keyboard in 32 mode type in the following:**

```
POKE &HFF04, (PEEK(&HFF03)
AND &HFE)
POKE &H10D, &HBF:POKE &H10E,
&H20
POKE &HFF03, (PEEK(&HFF03)
OR 1
```




# Shell sort routine

I HAVE written a database program for disc which when run loads all the entries in the first field containing the names. These are sorted into alphabetical order. My problem is that when new name is added, say 'Smith Mr. F' my routine passes down the entire list re-sorting them. This is very slow with over 200 entries. I would appreciate it if you would publish an amendment to my sort routine (below) or perhaps a better routine.

```
9010 FOR L=1 TO NE:FOR K=L TO NE
9020 IF NA$(L-1) <
NA$(K) THEN 9040
9030 T$=NA$(K):NA$(K)=
NA$(L-1):NA$(L-1)=T$
9040 NEXT K,L:RETURN
```

**THE routine you are using is very inefficient and this is why it is taking so long to resort after a new name is added. A much faster sort routine is known as the 'shell' sort which I've given below. In this case it may be better to simply search down the list of names once to find the position to 'insert' the name, rather than appending it to the end of the list and then resorting all the names.**

```
9000 A=1
9010 IF INT(2^A)
> NE THEN 9030
9020 A=A+1:GOTO 9010
9030 F=INT(2^A)-1
9040 F=INT(F/2)
9050 IF F=0 THEN 9170
9060 D=NE-F
9070 B=1
9080 A=B
9090 E=A+F
9100 IF NA$(A) > NA$(E) THEN 290
9110 B=B+1
9120 IF B>D THEN 9040
9130 GOTO 9080
9140 T$=NA$(A):NA$(A)=
NA$(E):NA$(E)=T$
9150 A=A-F:IF A<1 THEN 9110
9160 GOTO 9090
9170 RETURN
```

# Key notes

I AM writing a small program in Basic which plays a note after a key is pressed. The problem is that the ENTER key needs to be pressed after each choice.

Can you show me how to make the key being pressed be played immediately without having to use the ENTER key?

*Tudor Davies*
*57 Sunnyside Road*
*Weston-Super-Mare*
*Avon*

**THE solution is to use the INKEY$ function and the PLAY command together. INKEY$ will return the character pressed or an empty string if no character is being pressed. Hence you'll need something like the program below, of course you can enhance it to include playing 'sharps' etc.**

```
10 I$=INKEY$:IF I$
<'A' OR I$> 'G' THEN 10
20 PLAY I$:GOTO 10
```

# Slow strings

I HAVE 'inherited' a Dragon 32 and among the various programs is a database. I have been opening various files on this and I find that as I fill up the pages the program gets slower and slower. The program is in basic, and I'm told that the slowness is normal.

Do you know of a faster program that would overcome these problems?

*G. Holland*
*Llanfechell*
*Anglesey, Gwynedd*

**THE basic program most likely uses some form of strings to hold the data in RAM. As more information is entered it takes longer to search through all the strings to find the required item. Also basic needs to do 'garbage collection' occasionally when a large amount of string space is being used and this too will cause large delays.**

**The obvious answer is to try a commercial database, written in machine code which should not suffer from these problems and will probably offer you many more features. Some to consider are: the 'box' range from Harris Micro Software, *DataBase* from MST, and *DRS* from Grosvenor Software.**

# Tandy data transfer

I OWN a Dragon 64 and DragonDOS driving four drives using both Dragon basic and OS-9. I receive Tandy CoCo programs on discs and am in need of a program to read Tandy discs with DragonDOS.

I have tried swapping the directories from Tandy @ 17 to Dragon @ 20 but to no avail. If you or your readers are able to help it would really assist in maintaining interest in Dragons in this far flung corner.

*Dennis McQuade*
*62 Seaview Road*
*Paremata*
*New Zealand*

**UNFORTUNATELY, the disc formats of the CoCo and the Dragon differ quite considerably (the CoCo uses only 35 tracks, and the Dragon uses all 40). I have not seen a program to read Tandy discs directly (ie all files rather than simply sector by sector) on a Dragon, but I'm sure someone out there must have such a peice of software.**

**I know a large number of Dragon users also have access to Tandy CoCos, so the ability to transfer data via disc would obviously be useful. So if you have such a program (in basic or machine code) then send it to me, and I'll pass it on to Dennis and publish it on this page.**




WHAT happens when you take over your brother's adventure column? The first query I looked at was from one Eric Stockwell, asking Mike if he could possibly help to track down a tape written by his 'namesake' Pete Gerrard. Namesake? Blood brothers, dear boy. The tape in question was that which accompanied an early book of mine entitled *Exploring Adventures on the Dragon*, and if anyone else is interested in a copy (three adventures on one tape, although I must stress that they were written a couple of years ago) then they must write to Duckworth, The Old Piano Factory, 43 Gloucester Crescent, London NW1. Mention this column if you want, I know you lot, try anything for a discount. *(Duckworth's have a few copies of the book left at £6.95, and a good supply of the tapes. Call or write to the trade dept. on 01 485 3484 for further details, postage etc. -Ed.)* For calling me a 'namesake', Mr. Stockwell's tape is already on its way to him.

Following on from that, a very interesting letter from Andrew McBride of 109 Main Street, Lt. Harrowden, Nr. Wellingborough, Northants NN9 5BA. He apparently received a letter from Simon somebody-or-other who will probably get another mention before we're through, stating that all the text in the infamous *El Diablero* could be read by entering the following command:

```
FOR A=0 TO 32767:PRINT CHR$
(PEEK(A));:NEXT A
```

To quote Andrew: "Well, I thought if you could do this with *El Diablero* why not do it with any other adventure? It worked! All you have to do is reset it (don't worry if it cold starts)." A bit later on he continues "when you have listed it, it will look as if it has gone funny, but it's just the CHR$s. You can clear them by using an IF ... THEN statement."

Do we condone this sort of thing? We do, provided that it is only used in DIRE EMERGENCIES and you have given up all hope of solving the game. With this little track Andrew has managed to solve *Wings of War*, *Mountains of Ket*, *Temple of Uran* and *Syzygy*. Interesting stuff.

Interesting, because one Roy Williams has written to me with queries on *Mountains of Ket* and *Temple of Uran*. Spot the connection? Brother Mike has often mentioned the role he plays as a surrogate postman, so it looks as if that is another mantle that I will have to adopt. Roy also wants to know how to avoid getting slimed in *Final Mission*. Any offers, anyone? He says that he's got through the hole with the crowbar, the soap, the garlic, he's found the aerosol, which he used to get under the floor, but then can't open the further door due to aforementioned slime. All help gratefully received.

Help in exchange for help seems to be another common ploy used by you lot to get your names into this illustrious column. Richard Heath, residing at 66 Newcastle Road, Leek, Staffordshire ST13 5RU, has written in with a number of queries. He'd like to know what has to be done with the rock cakes in *Tanglewood*. They can be (SLIAF HCOUT RETFA) KRAHCA YB ENOTS DENRUT NEEB SAH HCIHW RETCARAHC YNA OT DEF. This backwards writing really slows down the typing speed, let me tell you. (*Funny, I was just thinking that too — Ed.*) Richard also wants to know how to stay alive in the radiation room past the second set of security guards in *Aquanaut 471*. Well, presumably you've found the seaweed and the mushroom. Anyone who's anyone and who is profoundly into vegetarian food will know that combining seaweed and mushroom produces a radiation vaccine pill which you can then eat.

And talking of *Aquanaut 471*, young Michael Edwards of Broomsoft, who has been mentioned in this column almost as often as Simon Whatsisname (and I must admit his name rings a bell from somewhere else as well ...) tells me how to cheat so that you don't have to play that ***** security robot game. Funny I don't remember reading that on the wrapper ... but anyway, load the game, and when you've started press RESET. Type:

```
POKE &H2911,&H7E
POKE &H2912,&H29
POKE &H2913,&HB2
EXEC 9744
```

Type QUIT to restore the position of any objects that you might have left lurking around in the game somewhere. That should also help Glenys Stephens, who in true grovelling style writes GREAT COLUMN in enormous letters (hope your ears burn, big brother) and then sends in please for aid. Help sheets should have reached you by now, and your last query on 'helpful snips' should by now be solved.

But before we leave Richard Heath entirely, having helped him out, he is now willing to help you out with his solutions to *Trekboer*, *Sea Quest*, *Shenanigans* and *Juxtaposition*. As with all people who have their addresses published here, please send an SAE if you want a reply. Every letter I've encountered so far that's been sent to this column has remembered their SAE, and although it took some time you should all have had replies by now. Is it my fault that Wigan get dumped out of the FA Cup and the world is thrown into disarray? It is not. On the other hand, however, I've received one or two teeny weeny complaints from people who have written (enclosing SAEs) to people mentioned in this illustrious column and have not had a reply. Shame on you. If I can miss *Eastenders* to write a few letters, I'm sure you can.

Peter Keefe is going to be a thorn in the Gerrard side, I can tell. Queries in mega-abundance, one of which is worthy of mention here because it's cropped up a couple of times. How do you get rid of the snake in *Callixto Island*? Having found some mice you can then ECIM PORD and SELDDAP EKAT. I must admit the lad's handwriting caused a twinge of amusement at one point, since he seems (in *Demon Knight*) to be asking how to get exceedingly drunk in the portcullis. Shome mistake, shurely? Now, on taking over I received a wodge of letters full of queries, along with brother's extensive file of solutions and adventures. Two adventures that you have asked about were conspicuous by their absence from this file, the earlier mentioned *Final Mission* that Roy Williams was having trouble with, and now *Demon Knight*. No solutions, no maps, not even a hint, so once more I send out a plea for help.

Of this large number of queries, the program that received the most mention was *Syzygy*, and easily the most troublesome feature of that game concerned entering the transporter co-ordinates. If you've solved this then apologies for re-stating what to you will be obvious, but from the number of letters I've read then many, may people are having problems here. No backwards writing this time, either, so read at your peril. The co-ordinates for the transporter are as follows:

Planet 0-4-1-5
Emerald 2-7-3-0
Vader 1-6-0-3

For each place you must enter each number by pressing that number and at the end of the row you must REVEL LLUP. And if it doesn't work, blame James Bonfield, not me. Anyone who lives in a place called Sandy can never really be sure of anything.

Nicholas Hart wants to know about good graphic adventures, so let's start up a bijou pollette. Write and let me know what your favourite graphic adventure has been to date, and as well as telling Nicholas a thing or two it might also make one or two companies sit up and take note and try and produce something better.

As an update to Mike's last column, in which he mentioned the Dragon Magazine (Michael Edwardes, Andrew Hill, the latter contactable at 13 Parry Jones Close, Blaina, Gwent NP3 3NH). A small update from the purveyors of the rag, stating that a year's subscription is priced at £6.50 for twelve issues, and this apparently barley (sic) covers costs. A sample issue is available for 60p from Andrew. It would appear that the magazine is now over one year old, a mere upstart in comparison to DU, but then we've all got to start somewhere!

Do you know, I nearly made it. Almost at the end of the column, no mention of Simon Thingummy, and what happens? Letters bombard me from all sides, raving about Dragon owners who use Prestel and play *Shades*, hints for *Tanglewood* and *Aquanaut 471* (and the aforementioned Shades: does this mean his future's so bright he's gotta wear shades?), and desperate please for me to mention yet another Dragon magazine. Okay, Hargrave, have it your way.

I mention these rival magazines as a lion might talk about a tabby — appropriately, this one is called Dragon's Tail, and a year's sub costs £7.50. Since I haven't seen a copy yet (we refer to this as The Hint) you will have to send off your pfennigs on spec, but the person to write to is Julian Cogdell, Dragon's Tail, 34 Shooters Drive, Nazeing, Essex EN9 2QD. Ask 'em if they sell sample copies if you just want one to look at.

Perhaps someone will send me a copy someday.

Hope that you enjoyed my first flight into Dragonland. It's always difficult taking over an established column, but we'll see what can be done. Next month I'll be looking at (among other things) half a dozen adventures that people have sent in for review, plus of course any other games that might crop up between now and then. Your letters are of course always welcome. I can already spot the troublemakers and will deal with them accordingly (you know your names, you guys) and the rest of you too — just keep 'em coming.

## Adventure Contact

To help puzzled adventurers further, we are instituting an Adventure Helpline — simply fill in the coupon below, stating the name of the adventure, your problem and your name and address, and send it to Dragon User Adventure Helpine, 12/13 Little Newport Street, London WC2H 7PP. As soon as enough entries have arrived, we will start printing them in the magazine.

Don't worry — you'll still have Adventure Trial to write to as well!

**Adventure** ..........................................

**Problem** ..........................................

..........................................

**Name** ..........................................

**Address** ..........................................

..........................................

# CLASSIFIED ADS



HERE'S MY CLASSIFIED AD.
(Please write your copy in capitals on the lines below)

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

Name ..........................................

Address ..........................................

.......................... Tele ..........................

Please cut out and send this form to: Classified Department, Dragon User, 12 —13 Little Newport St, London WC2H 7PP.

# Three letter cards

***Gordon Lee** shuffles uneasily while he waits to see what the cards say*

WOULD-be competitors often write to say that they would like to enter the competition but feel that they lack the necessary experience to tackle the problems. In fact, apart from the possibility of winning one of the prizes, one of the spin-offs of attempting the competitions is the opportunity to devise original programs, starting with the interpretation of the problem, finishing with the final program, which should, you hope, provide the correct answer. Many readers possibly use their Dragons either only for loading pre-recorded programs, or for copying listings from the magazine. In fact, most of the competitions require only a logical approach plus a knowledge of some simple Basic commands. Even the inevitable mistake can help to pinpoint a particular pitfall, and the debugging of a program can prove particularly instructive. Also the experience gained can have an application in other areas of programming.

This month, this column is devoted to all you beginners, so experienced programmers may like to jump to the competition at this point.

At the heart of all programming is the management of variables. Whether it is in the creation, examination, comparison or alteration of variables, a knowledge of them is essential for successful programming. Take a look at a Basic listing, and you will find that most program lines contain a variable in some form or another. Note that these variables come in two flavours — numeric (used to store numbers) and string (use to store text or other characters). The string variables are indentifiable by the dollar sign.

Quite often variables can be used to 'mimic' actual objects or procedures. For example, if we wished to create a pack of cards, we could use listing 1. If this is typed in and run, the command PRINT PACK$ will result in a 104 character string containing each of the playing cards, in order. Note that each card is represented by *two* characters in the string, for example, 3H, JS, or AD, for the three of hearts, jack of spades and ace of diamonds respectively. From this it is easy to select a card at random, for example by typing in X=RND(52)*2-1:PRINT MID$(PACK$,X,2)

The value X denotes the position in the string at which the random card is to be found, and must obviously by an odd number — hence the routine of multiplying by two and subtracting one. Thus, if this character and the next in the string is printed it will result in a pair which indicates a playing card value. An extension of this technique can be used to 'shuffle' the cards. The computer does this by starting with an empty string (SHUF$=" "), and adding to this a series of 52 cards selected at random from the PACK$ string. To

**Listing 1**

```
10 PACK$="":SUIT$="CDHS":RANK$="A23456789TJQK"
20 FOR S=1 TO 4:FOR R=1 TO 13
30 PACK$=PACK$+MID$(RANK$,R,1)+MID$(SUIT$,S,1)
40 NEXT:NEXT
```

**Listing 2**

```
50 SHUF$="":FOR N=52 TO 1 STEP-1:X=RND(N)*2-1
60 SHUF$=SHUF$+MID$(PACK$,X,2)
70 PACK$=LEFT$(PACK$,X-1)+MID$(PACK$,X+2)
80 NEXT
```

**Listing 3**

```
  5 DIM HAND$(4)
 90 C=1
100 FOR H=1 TO 4:HAND$(H)=HAND$(H)+MID$(SHUF$,C,2):C=C+2:NEXT
110 IF C<104 THEN 100
```

**Listing 4**

```
 50 N$=PACK$:T=1
 60 SHUF$="":FOR C=1 TO 26
 70 SHUF$=SHUF$+MID$(N$,52+(C*2-1),2)+MID$(N$,C*2-1,2)
 80 NEXT
 90 PRINT:PRINT SHUF$;" (";T;")"
100 IF SHUF$=PACK$ THEN 120
110 T=T+1:N$=SHUF$:GOTO 60
120 END
```

prevent the same card being selected twice, as a card is picked it is removed from the PACK$ string at line 70. Listing 2 lines 50 to 80 perform this shuffling operation.

Having done the shuffling, we might wish to deal four hands of cards. If you add listing 3 (line 5, and lines 90 to 110) the cards are dealt into the dimensioned array HAND$(). Of course, it is perfectly possible to combine listings 2 and 3 and feed the randomly selected cards directly into the array without the need for a separate 'shuffle', but the listings as given more closely imitate the actual procedure using actual cards.

Another type of shuffle is the riffle shuffle. Here a pack is divided into two equal halves and the cards are riffled so that they fall alternately (at least, that is how it works in theory!). In practice, the cards are not likely to oblige and interlace so exactly. Curiously, a series of perfect riffle shuffles will restore the pack to its original order. If you replace lines 50 to 110 with listing 4 the computer will now perform a series of riffles until the order of the pack returns to its original state. Note that in performing the shuffle, when the pack has been divided the card which is allowed to fall first is the one from the top half of the pack. You will be surprised to find that, by an odd coincidence, fifty-two shuffles are required to restore the order. If two cards are removed, then only eight shuffles suffice!

## Competition

For this month's competition we need to simulate some packs of cards, not with the conventional suits and values, but bearing the letters of the alphabet from A to Z. I have three piles of cards each comprising a complete alphabet. They are placed in a row, face up, with the cards in alphabetical order, A on top and Z at the bottom. I cut each of the packs several times, completing the cut on each occasion. Note that this does not alter the order of the cards, just their relative positions from the top of the pack. For example, the topcards might read YJB. If I then remove one card from each pile I will get a new sequence, and by repeating the procedure twenty-five more times, will work my way through the packs. In so doing, I will arrive at a series of three-letter combinations, for example, YJB, ZKC, ALD, BME, CNF, DOG, EPH etc. Occasionally, I may produce a word, such as DOG, and in some cases it is possible to find two or more words in the same 'run' of cards. ATE, for instance, will produce PIT, and NOT will produce STY. Some 'triplets' are TWO, LOG and RUM, or BUS, HAY and PIG.

Can you find any sequences of more than three words?

## Prize

Peter Gerrard is taking over *Adventure Trail* this month from his overworked brother Mike. Peter is uniquely suitable for this demanding post since he is not content simply to carry on the family name in the adventure business, he has to go writing books about it as well.

Pete's book, published some time ago by Duckworth's (and still — just about — in print) came out alongside a cassette of the same subject, *Exploring Adventures on the Dragon* on the (you guessed it) Ducksoft label.

Details on where to track down the book and the tape are given by Peter in the *Adventure Trail* on page 28, but Duckworth's have kindly awarded us an unspecified number of tapes for prizes for this month's competition — so let's frighten 'em.

## Rules

So show us your hand. You know the rules: suss out the strategy, plan the program, send us a listing of your solution (no cassettes please) and any notes you want to include. Mark your envelope with the words JANUARY COMP ... (*No, you twerp*) sorry *MAY* COMPETITION, as well as Dragon User's name and address (don't forget to include your own name and address) and send it to us.

Our tie breaker for the May competition is to answer the question "What's a Duck Worth ?", using your wit, skill and judgement. No packet tops required.

## February winners

In contrast to the March competition (also popularly known as the 'January' competition, thanks to the efforts of a sub editor sitting not a thousand miles from this workstation), where entries are pouring in on fag packets (OK, opened-out fag packets) and the backs of blank cheques, the entries for the FEBRUARY competition have sailed in, slow and stately, wading low in the water under their massive freight of program analysis, wit and footnotes. We're offering a set to the *Encyclopaedia Brittanica* as a follow-on.

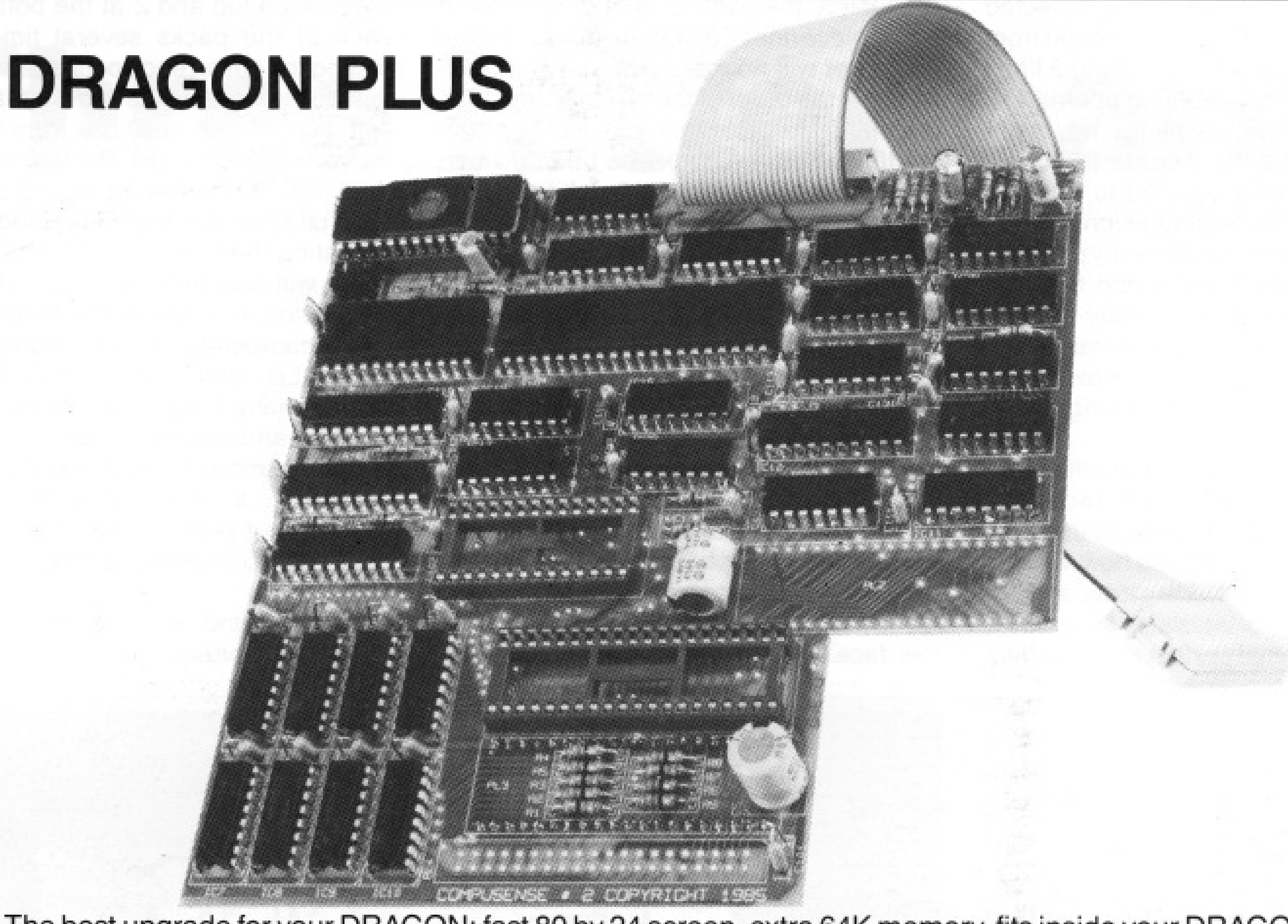
This month's winners, budding novelists all, are R. H. Wilson of Basingstoke, G. R. Barber of Sutton Coldfield, S. A. Siddiqui of Chiswick, T. Fawcett of Hendon, Phil Sapiro of Liverpool, H. Grandy of Sheffield, Mark Towlson of Nottingham, M. Philips of Bristol, Mike Dunn of Leicester, Alan Thomas of Staplehurst, Greg Cole of Crawley and Paul Weedon of Wooton-Under-Edge. Copies of Blaby's *Ruby Robba* are in the post, and we have a few spares here. No begging letters please! They will be going to deserving causes (such as the Editor's Magic Bottomless Box for special services and bereaved prizewinners.) Blaby's list of games is now available from R & AJ Preston Software. Call (0656) 880965 for more information.

And an emergency apology: Entrant D. J. Gray of Middlesborough didn't get his name check in last month's winners. Gordon wants to know why, and says his entry was first class and one of the most original. Sorry, D. J.. Your game is on its way.

## Solution

Up to 158 primes were counted.

# DRAGON PLUS




The best upgrade for your DRAGON: fast 80 by 24 screen, extra 64K memory, fits inside your DRAGON. Hundreds of satisfied users. Installation service available. Requires 64K DRAGON (upgrade service for DRAGON 32K — CALL US). Especially recommended for OS-9 users — get the 80 * 24 screen AND save the 6K overhead of the 51 column screen. Use Stylo and C without having to reboot!

| | | | |
|---|---|---|---|
| DRAGONPLUS | — £103.50 | EDIT + Cartridge/Disk for DRAGONPLUS | £14.95 |
| FLEX DRIVERS | — £5.75 | OS-9 Drivers | £5.75 |

## FLEX SOFTWARE

Get the best out of your DRAGON with FLEX operating system. 51 by 24 screen standard. 48K free memory for your programs. Includes the best 6809 assembler available for the DRAGON. DBASIC lets you run DRAGON BASIC programs — includes cheque book program. Requires 64K and disc drive.

| | | | |
|---|---|---|---|
| FLEX Editor/Assembler/DBASIC | — £99.99 | DYNACALC | — £49.50 |
| SPE word processory spell check | — £65.00 | RMS Database | £49.50 |

***** SPECIAL OFFER — complete package — £199.00 *****

FORTH 83 for FLEX - complete implementation on two disks with source £25.00

## The Best Modem software for the DRAGON 64

CHAT COMBO (chat plus 2, XMODEM, CHAT/Viewdata £45.00

CHAT/Viewdata access PRESTEL (cassette/DragonDos/FLEX) £14.95

## DRAGON HARDWARE — SPECIAL OFFERS

| | |
|---|---|
| EPROM PROGRAMMER Mark 1 (KIT) for DRAGON 32 reads/programs up to 2764 | £25.00 |
| EPROM PROGRAMMER Mark 2 (Assembled) reads/programs up to 27256 | £55.00 |
| MODEM/RS-232 &I/O interface for Dragon 32/64 (uses 6551 ACIA and 6522 VIA) (bare board with instructions) | £5.95 |
| CASE for RS-232 | £1.95 |
| FLOPPY DISKS (Certified double denisity double sided) | 10 for £9.95 |

FREE DISKETTES WITH DRAGON DISK DRIVE SYSTEMS

## DRAGON/PC Convert £24.95

Now you can convert your Dragon BASIC programs and data to run on the IBM or compatable (eg Amstrad) Personal Computer PC, XT or AT.

Dragon Basic is very similar to the standard BASICs used on PCs. The hard part is transfering your programs and data to the PC's disk. DRAGON/PC Convert is a program for PCs which reads DRAGON-DOS format disks and copies data and programs to standard MS-DOS format.

Buy an AMSTRAD PC — get DRAGON/PC Convert FREE
From £449.00 (£516.35 inc VAT) plus £9.00 carriage

FAST MAIL ORDER — PRICES INCLUDE VAT — PLEASE ADD £2.00 POST & PACKING





**COMPUSENSE**
**68A WILLOUGHBY LANE, LONDON N17 OSP**
**Telephone: 01-885 3300 (2 lines)**

**FAST MAIL ORDER**

**COMPUSENSE** 